Project Session 01:
Project Planning and Structure

You Already Know

Before we begin, let's recall what we have covered so far:

- Agile
- Git

A Day in the Life of a Full Stack Developer

Meet Mr. Bob. He is a scrum master working for Zest Corp.

A Day in the Life of a Full Stack Developer

Meet Mr. Mark. He is the team's product owner.

Hi Bob! We want to increase our product sales, so I was wondering if you could build a user-friendly e-commerce web application that allows users to find all their requirements on a single page.

Sure, Mark. I'll get started.

A Day in the Life of a Full Stack Developer

Now, Mr. Bob is contemplating Mr. Mark's needs and is grappling with the idea of creating an e-commerce web application that encompasses all the options on a single page.

A Day in the Life of a Full Stack Developer

He has got the below requirements from the Product Owner:

Project Planning and Management: Utilize Agile methodology for project planning to ensure flexibility, continuous improvements

Front-End Development: Implement dynamic and responsive frontend of the e-commerce website using Angular and Node.js

Back-End Development and Database Management: Develop the backend logic using Spring Boot and Java and use MySQL to manage the database and efficient data storage and retrieval

Version Control and Collaboration: Use Git for version control to manage code changes and track the project's progress

DevOps Implementation: Use Docker for containerization, Jenkins to automate the workflow of the application, and deploy it on AWS for scalability, reliability, and security.

Functional Requirements

Mark has also provided the following functional requirements for the e-commerce web application:

Functional Requirements:

1. **Agile Methodology for Project Planning**:
   - **Requirement**: Implement Agile methodology to manage project planning and development.
     - **User Stories**: Break down the project into user stories and tasks. Include all the functionalities for Users and Admin. Moreover, provide the information of complete product workflow.
     - Admin user stories will include the stories related to the product details, user management, inventory management, payment, and orders.
     - End-user user stories will include the stories related to the User functionalities, such as profile details, orders, Wishlist, payment, and product page.
     - **Sprint Planning**: Conduct sprint planning sessions to define tasks for each sprint.

Functional Requirements

2. Front-End Development with Angular and Node:

- **Requirement:** Use Angular and Node.js to develop the front end of the web application.
- **UI Components:** Develop reusable UI components using Angular.
- **Service Integration:** Integrate with backend services using RESTful APIs.
- **Routing:** Implement client-side routing for navigation.
- **Form Handling:** Create and manage forms for user input.

3. Back-End Development with Spring Boot, Java, MySQL/MongoDB:

- **Requirement:** Use Spring Boot and Java to develop the back end, with MySQL/MongoDB for data storage
- **API Development:** Develop RESTful APIs for client-server communication
- **Database Integration:** Implement database connections and CRUD operations with MySQL/MongoDB
- **Authentication and Authorization:** Implement user authentication and role-based access control
- **Business Logic:** Implement business logic and services

Functional Requirements

4. Version Control with Git and GitHub:

- **Requirement:** Use Git and GitHub for version control and collaboration
- **Repository Management:** Create and manage repositories on GitHub
- **Branching Strategy:** Implement a branching strategy for feature development and releases
- **Pull Requests:** Use pull requests for code reviews and merging changes
- **Continuous Integration:** Set up continuous integration workflows to automate testing and deployment

Non-Functional Requirements

Mark has also provided the following non-functional requirements for the e-commerce web application:

Non-Functional Requirements:

1. **Performance**:
   - The back end should handle at least 1000 concurrent users without performance degradation.

2. **Scalability**:
   - The system architecture should support horizontal scaling to handle increasing loads and accommodate growing data.

3. **Security**:
   - All sensitive data should be encrypted in transit and at rest.

4. **Usability**:
   - The application should have an intuitive and user-friendly interface.
   - Provide comprehensive documentation and help resources for end users.

Non-Functional Requirements

5. **Reliability**:
   - The system should have an uptime of 99.9%.
   - Implement automated backups and disaster recovery mechanisms.

6. **Maintainability**:
   - Code should follow best practices and be well-documented to ease maintenance.
   - Implement automated testing to ensure code quality and reliability.

7. **Compliance**:
   - The application should comply with relevant industry standards and regulations (e.g., GDPR, HIPAA).

8. **Accessibility**:
   - The web application should be accessible to users with disabilities, following WCAG 2.1 guidelines.

A Day in the Life of a Full Stack Developer

Bob has received the below high-level requirements from the product owner:

1. Create an admin dashboard where the admin user can create, update, and delete a product detail
2. Build the functionality for admin user to maintain users, orders, payment, and shipping
3. Create an end-user dashboard where the customer can search for a product, filter the search based on their requirements, add the product to the cart, and make a purchase
4. Create a database with either MySQL or MongoDB
5. Automate the deployment of the application with Jenkins
6. Deploy the application on AWS

A Day in the Life of a Full Stack Developer

Bob now plans his activities to create a web application following his research.

He decides to do the following activities:

- Create user stories for Epics of web app for the end user and admin Dashboard
- Plan sprints using JIRA
- Create projects using angular framework
- Sync projects using GitHub repositories

In this lesson, we will learn how to solve this real-world scenario to help Bob complete his task effectively and quickly.

Learning Objectives

By the end of this lesson, you will be able to:

- Develop user stories for admin dashboard epic and end-user Epic
- Execute stories based on the planning done for the sprint
- Create a Web App Project for the end user using Angular CLI
- Implement admin Dashboard Project code using git on GitHub
- Implement Web App Project code for end user using git on GitHub

User Stories: Web Admin Dashboard Epic

Admin Dashboard

In this project, various modules will be managed by admin, such as:

Listing the products for end-user

Managing the orders

Managing the users

Authentication Story

The admin user should be able to login to the admin dashboard using admin user credentials.

- The admin user should perform identification authentication using email and password with captcha verification to access admin site at the backend.
- A web page with text fields for username and password with a login button to log in to the system is necessary.

Authentication Story

The username and password should not be empty. Password must be at least 6 characters.

Username and password are needed for authentication, and these fields should not be empty.

Password should be minimum of 6 characters.

User should be navigated to the dashboard page after logging in.

Product Story, Product List, and Product Details

The admin user should be able to manage products and add details, such as:

Price
and discount

Material and
size description

Availability of
quantity

Product usage
video tutorial

Product Story, Product List, and Product Details

The product screen should be divided into two parts for the description of each product.

Products List	Products Description
Product 1	**Product 1**
Product 2	Description:
Product 3	Size:
Product 4	Price:
.....	Availability:

Product Story, Product List, and Product Details

At least two product images for each product must be uploaded, including its price, size, and availability details.

Products List

Product 1

Product 2

Product 3

Product 4

.....

Products Description

Product 1

Description:

Size:

Price:

Availability:

Product Image: 

Managing Users

The admin user should be able to access users' data and monitor newly registered users and frequently visited sections by the users.

Managing Users

By default, the user list page should be empty.

Then, users can be filtered using user filtration criteria. After filtration, the web page is divided into two parts.

The first part shows a list of users, and the second part shows user descriptions such as date of registration and activity level.

Managing Users

User List

Filter by

Frequently active
Last active
Date of registration

User 1

User 2

User 3

User 4

.....

User Description

User 1

Date of registration:

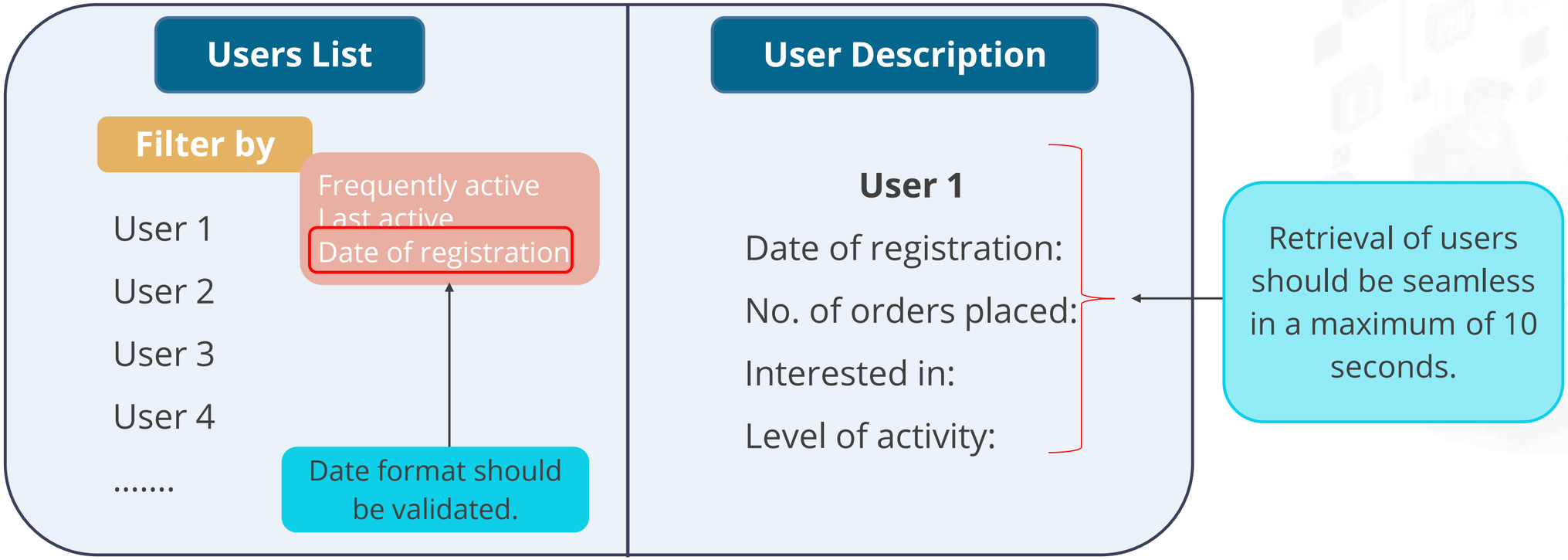
No. of orders placed:

Interested in:

Activity level:

Managing Users

The date format should be validated before sending requests for users' lists.



Order Story

The admin user should be able to view new orders as well as finished orders.

They should be able to manage order status, such as:

Availability of ordered products

Order transition information

Order Story

The web page is divided into three sections.

New Orders	Ongoing Orders	Halt State
Order 45	Order 39	Order 28
Order 46	Order 40	Order 30
Order 47	Order 43	Order 35
Order 48	Order 44	Order 41
.....	.....	.....

Order Story

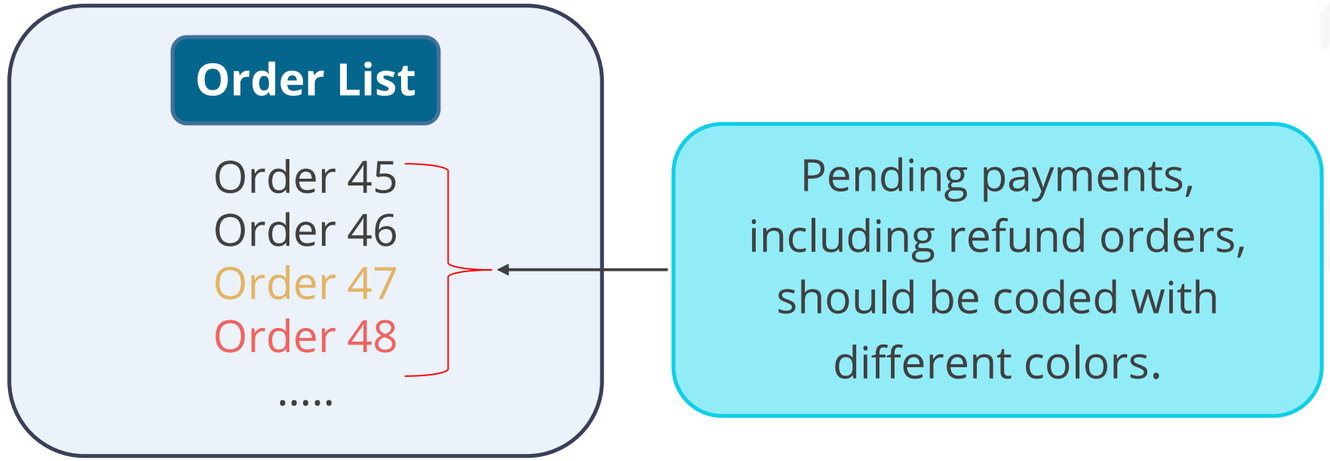
While processing the orders, the admin user should validate the availability of the products in the inventory.

New Orders	Ongoing Orders	Halt State
Order 45	Order 39	Order 28
Order 46	Order 40	Order 30
Order 47	Order 43	Order 35
Order 48	Order 44	Order 41
.....	.....	.....

Check for availability

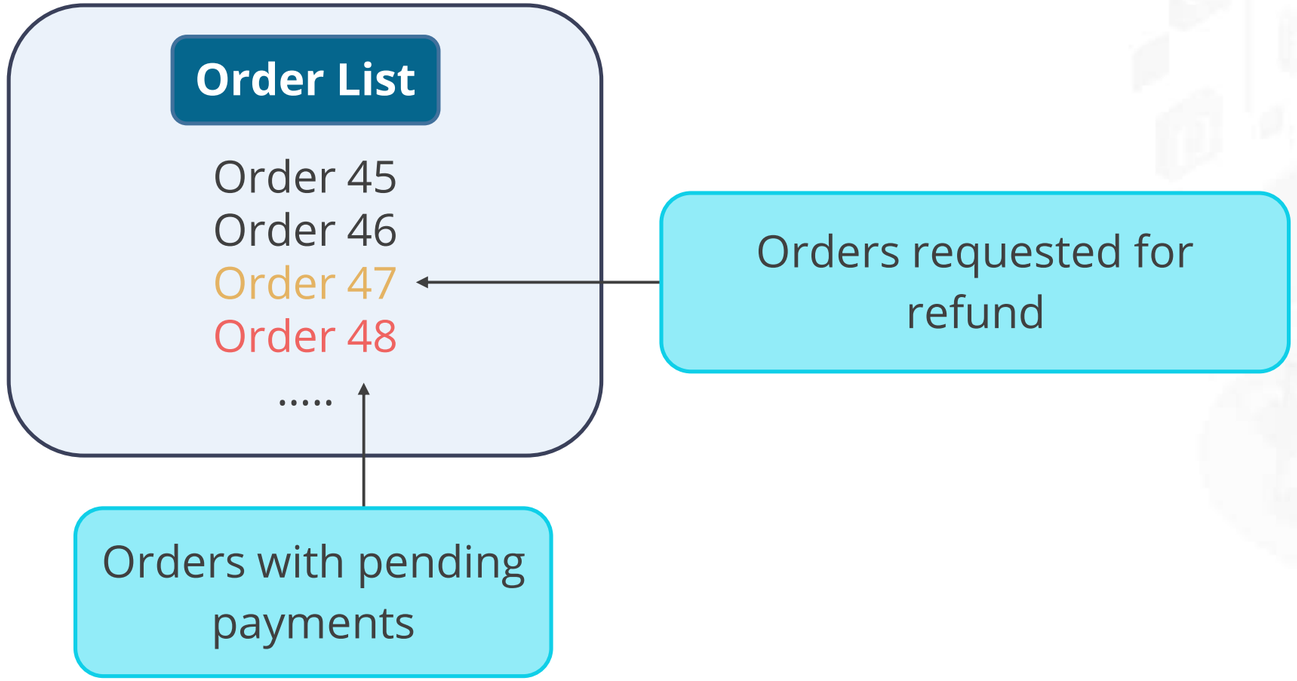
Payment Story

The admin user should be able to monitor and view the list of orders whose payment status is pending.



Payment Story

On the web page, a list of all pending payments, including refunds, should be color coded for easy identification.



Shipping Story

The admin user should be able to view orders which are either ready for shipment or in transition.

They should be able to update the order's delivery date and current location for end-user reference.

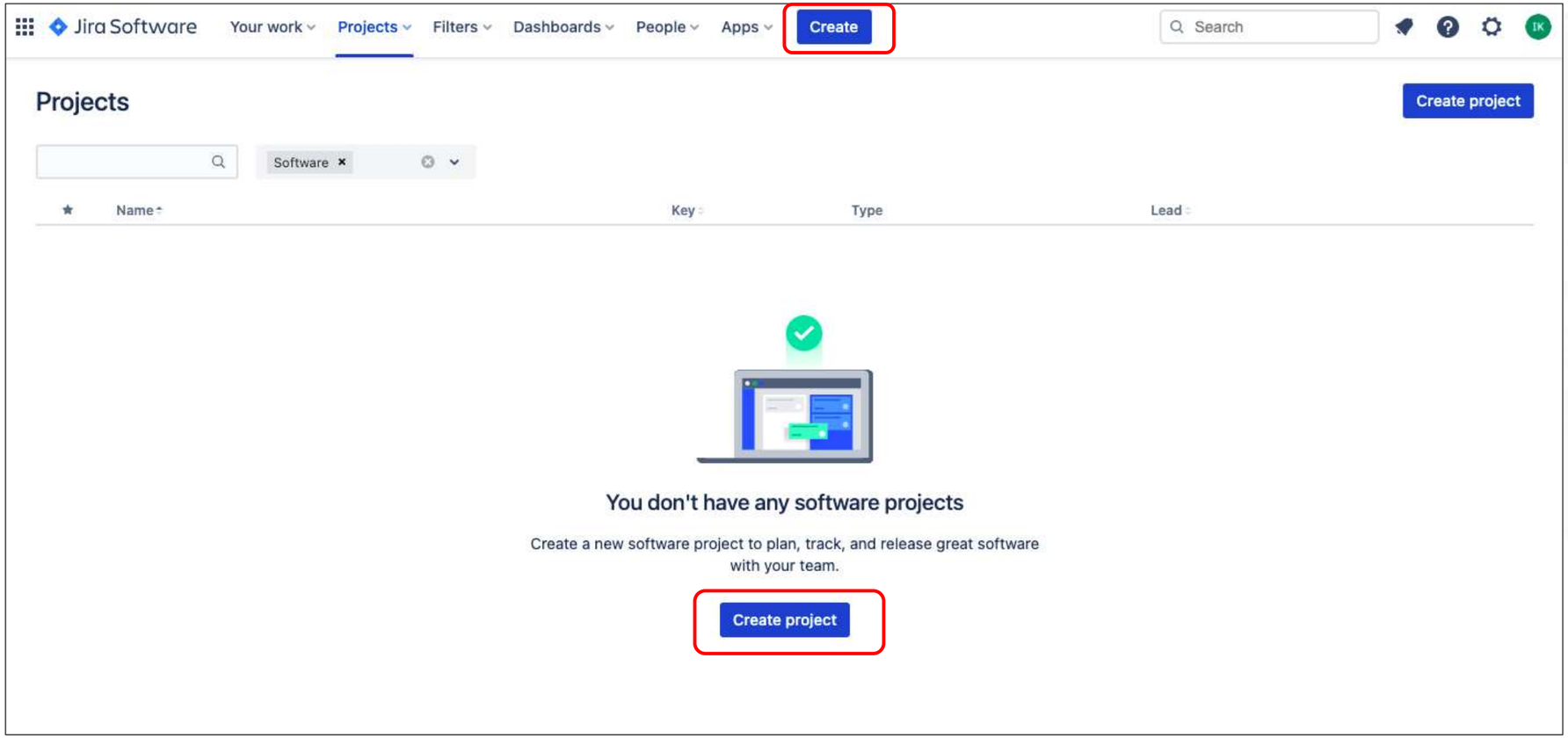
Orders	Orders	Orders
Order 45	Order 39	Order 28
Order 46	Order 40	Order 30
Order 47	Order 43	Order 35
Order 48	Order 44	Order 41

Orders that are on their way to their respective addresses.

By selecting any order, the admin user should be able to update the order's delivery date and current location.

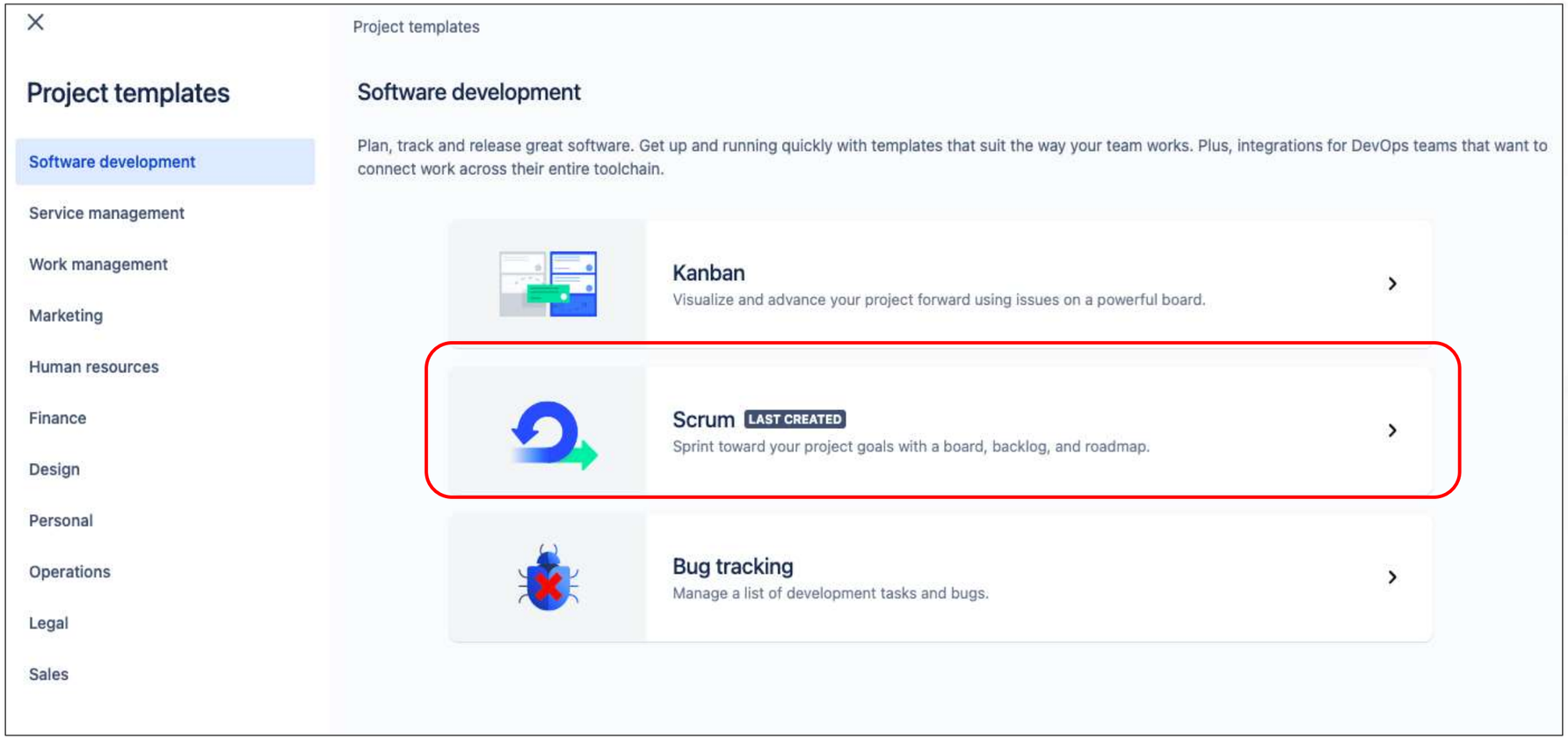
Create Project in JIRA

Once an account has been created in JIRA, the first step is to create a project that will be managed using Agile.



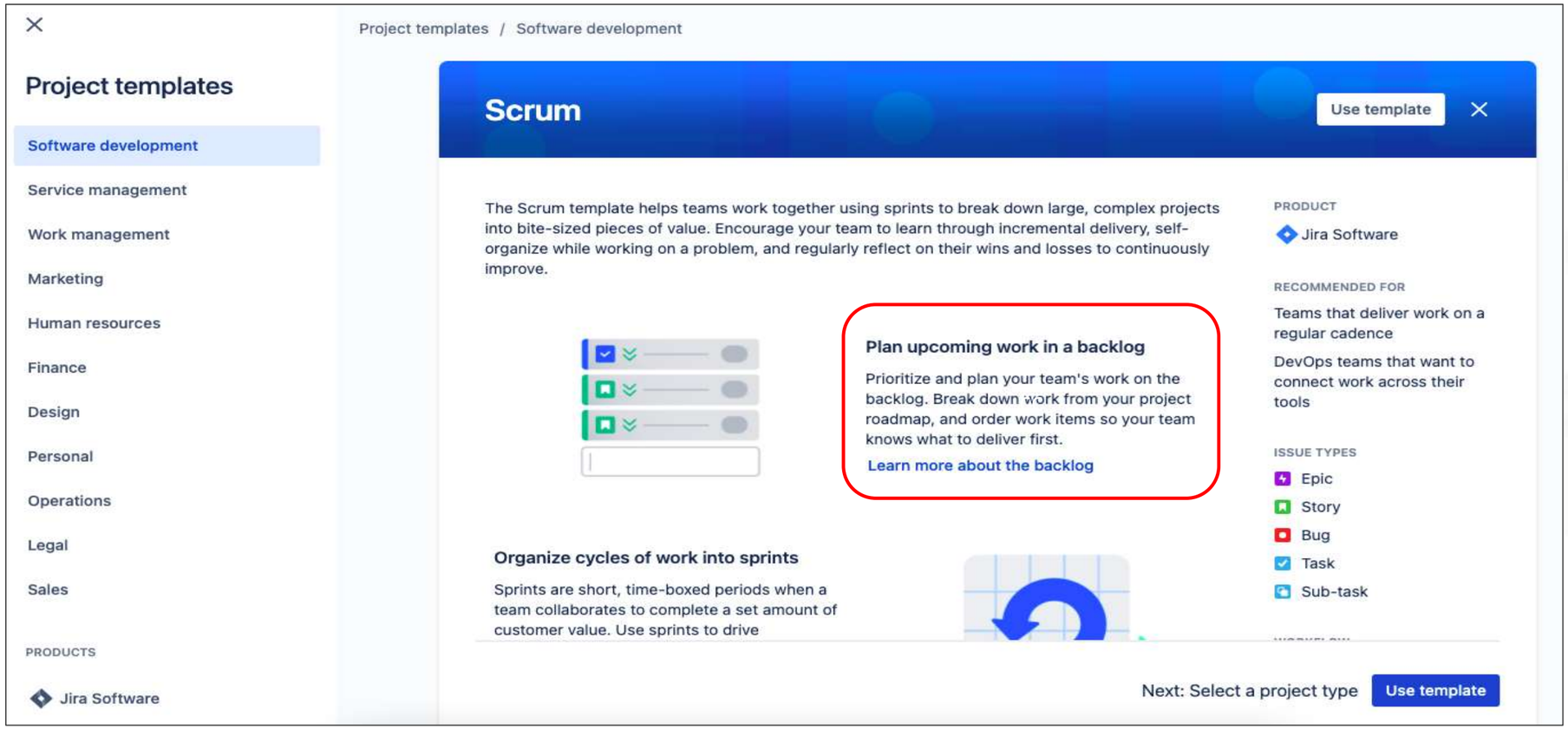
Different Project Options

Click on Create Project and choose Scrum as the preferred methodology.



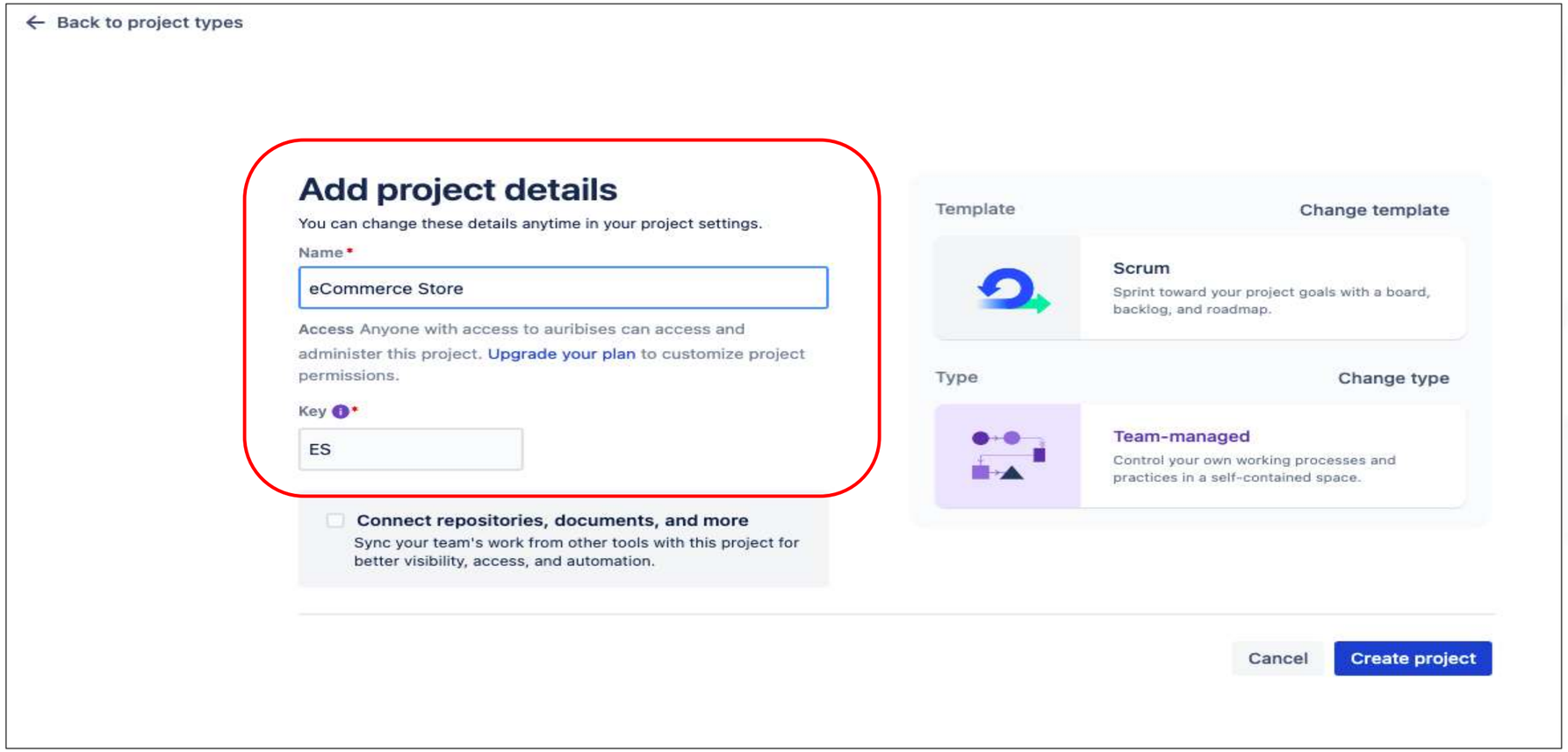
Scrum Template

With the Scrum project theme, the user will be able to plan and create Sprints and break down a complex project into small projects.



Project Details

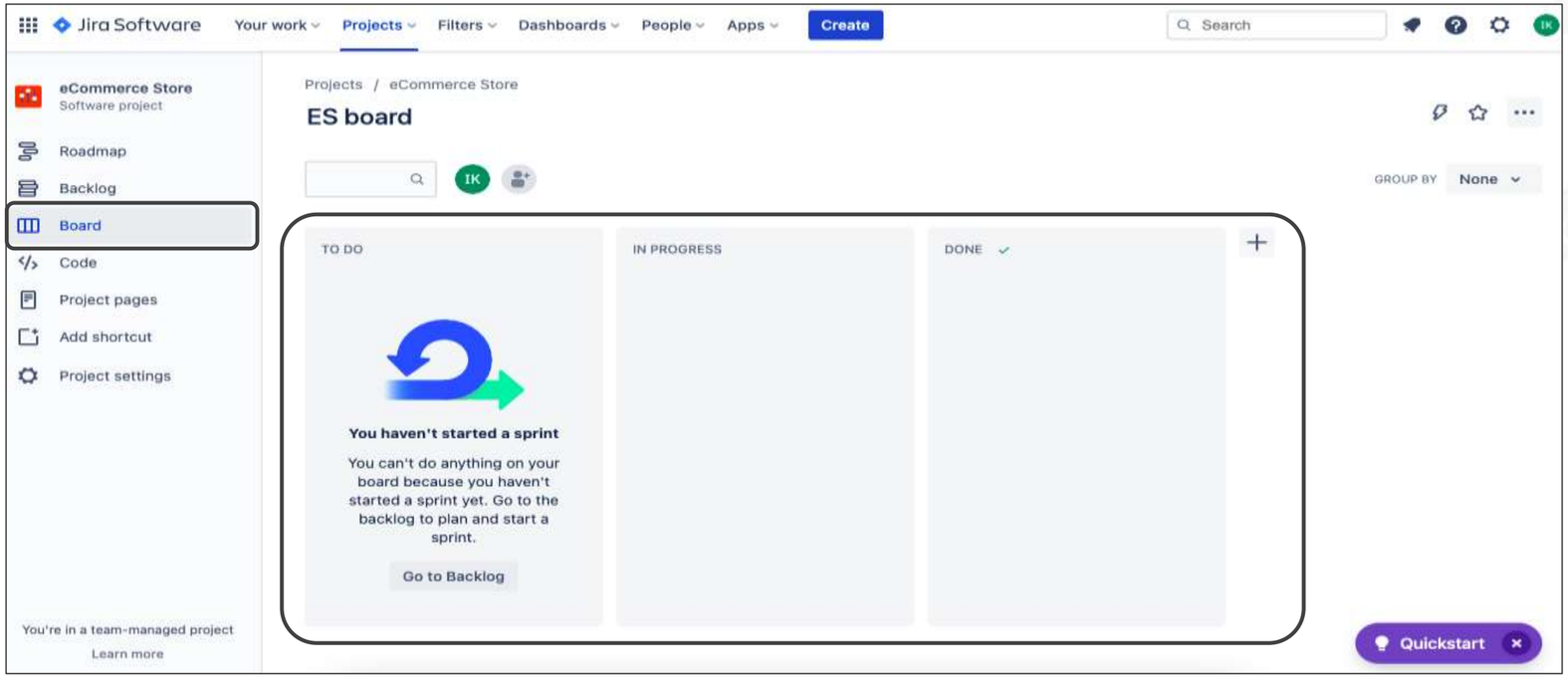
Next, the user will add details of the project, including the project's name. A key will be auto-generated for the project, which can be changed as required.



Project Home Page

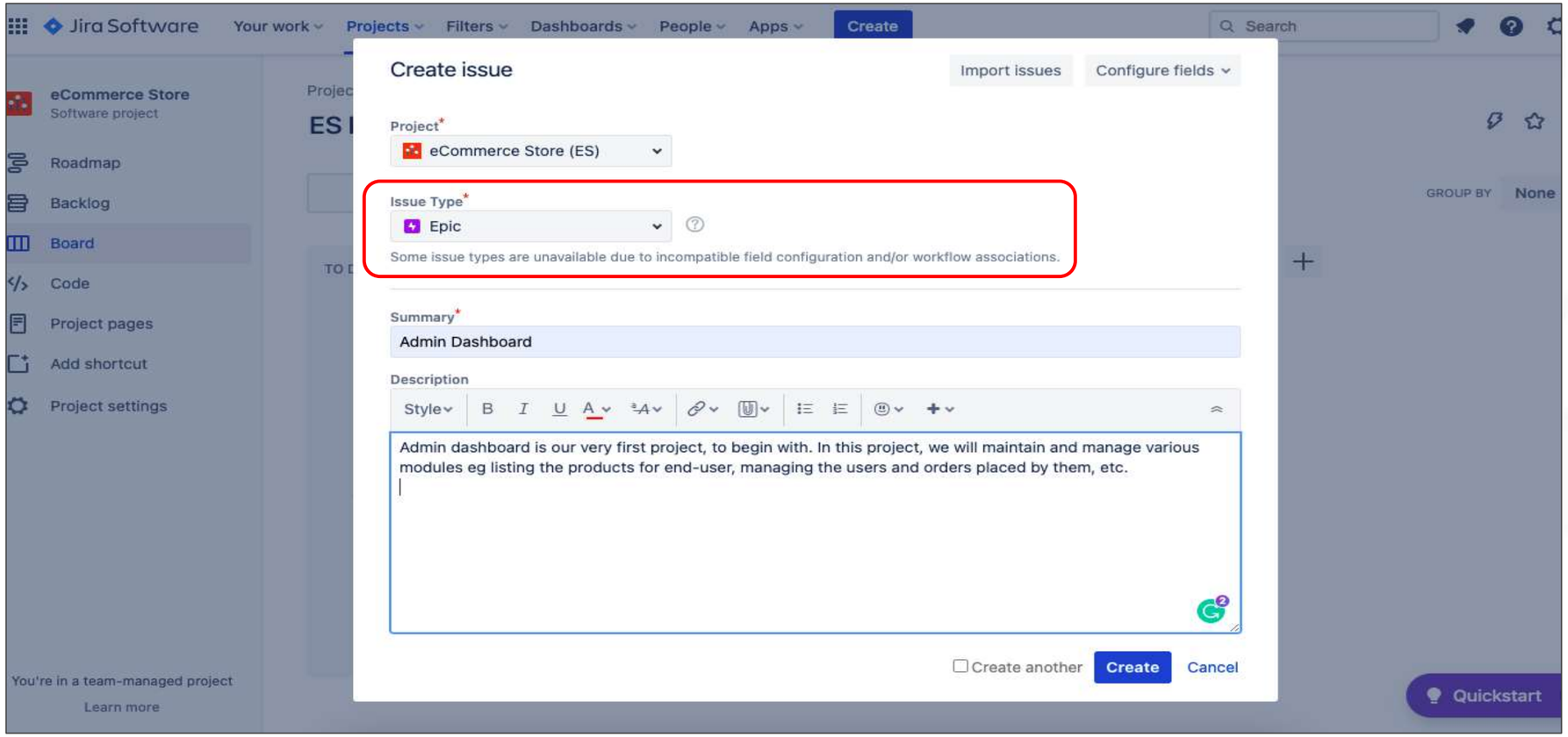
After creating and adding the project, the user will be redirected to the home page with Boards options.

In Boards, the task can be grouped as **To Do**, **In Progress**, and **Done**. If required, any custom board can also be added.



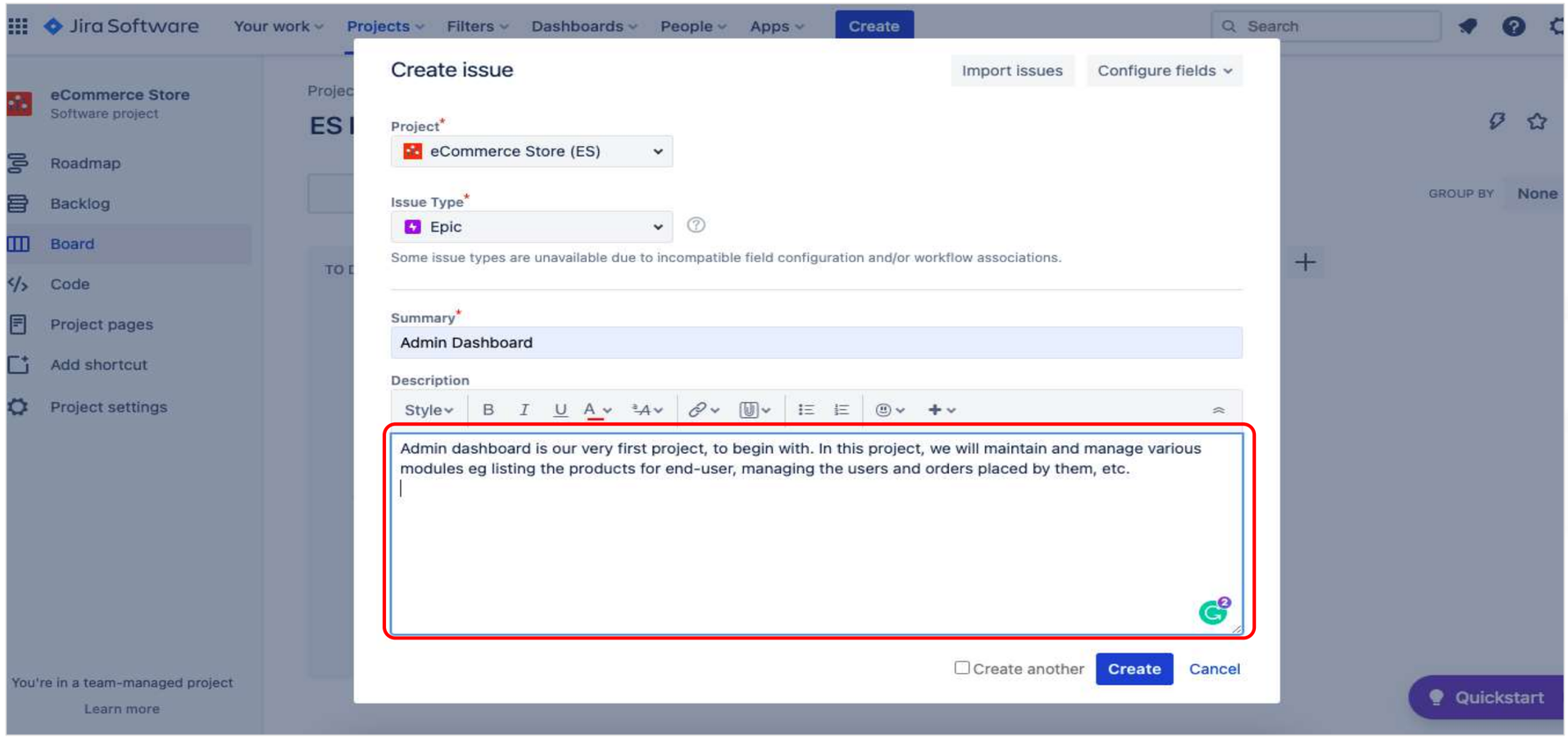
Create Admin Dashboard Epic

The next step is to create Epics. Click on the create button on the project home page and an option to create an Issue will be displayed. Select **Epic** as the issue type as shown below:



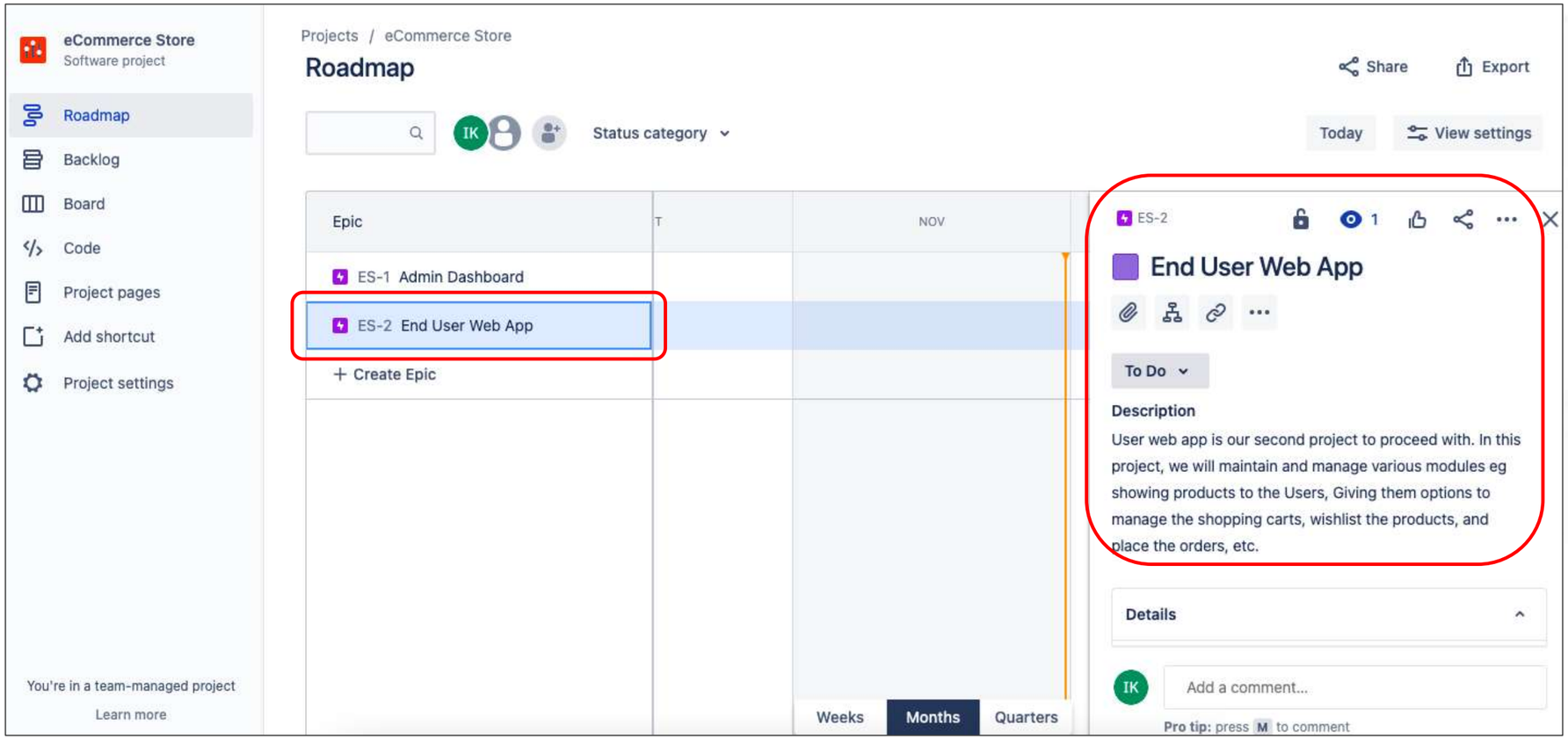
Create Epic

The issue will be Epic, Story, Task, or a Bug. Select Epic in Issue Type and write a summary and detailed description and similarly create the User Dashboard Epic and add the description for the same.



Road Map

Added Epics will be viewable by the user in the Projects Roadmap. When selecting an epic, its summary and description should be visible.



Admin Dashboard User Stories

Here is the list of the user stories for the Admin Dashboard Epic:

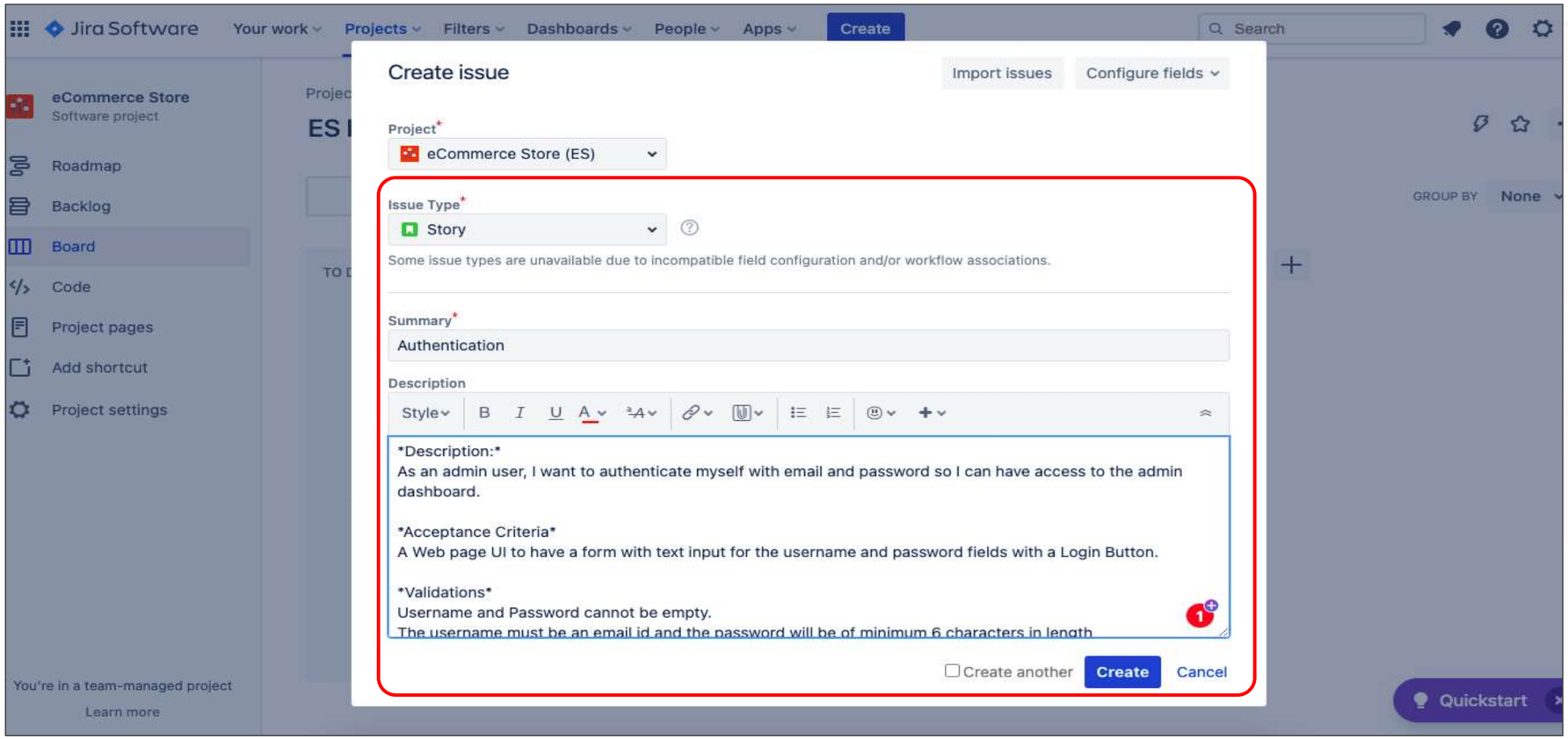
Role	User Story	Description	Acceptance Criteria
Admin	As an Admin, I want to manage user accounts so that I can maintain control over the users of the e-commerce platform.	Admin can create, read, update, and delete user accounts.	• Admin can see a list of all users. • Admin can add a new user. • Admin can edit user details. • Admin can delete a user account.
Admin	As an Admin, I want to manage product details so that I can ensure the product catalog is accurate and up-to-date.	Admin can create, read, update, and delete products in the catalog.	• Admin can add a new product with details like name, description, price, and stock quantity. • Admin can edit existing product details. • Admin can remove products from the catalog.
Admin	As an Admin, I want to manage inventory so that I can keep track of stock levels and update them as needed.	Admin can view and update stock levels for each product.	• Admin can see current stock levels for all products. • Admin can adjust stock levels manually. • Admin receives notifications for low stock levels.

Admin Dashboard User Stories

Role	User Story	Description	Acceptance Criteria
Admin	As an Admin, I want to manage orders so that I can oversee and process customer orders.	Admin can view, update, and process orders.	• Admin can see a list of all orders. • Admin can update the status of an order (e.g., pending, shipped, delivered). • Admin can cancel orders if necessary.
Admin	As an Admin, I want to manage payment transactions so that I can handle billing and ensure all transactions are processed correctly.	Admin can view and manage payment transactions.	• Admin can see a list of all payment transactions. • Admin can issue refunds for transactions. • Admin can handle failed or disputed transactions.

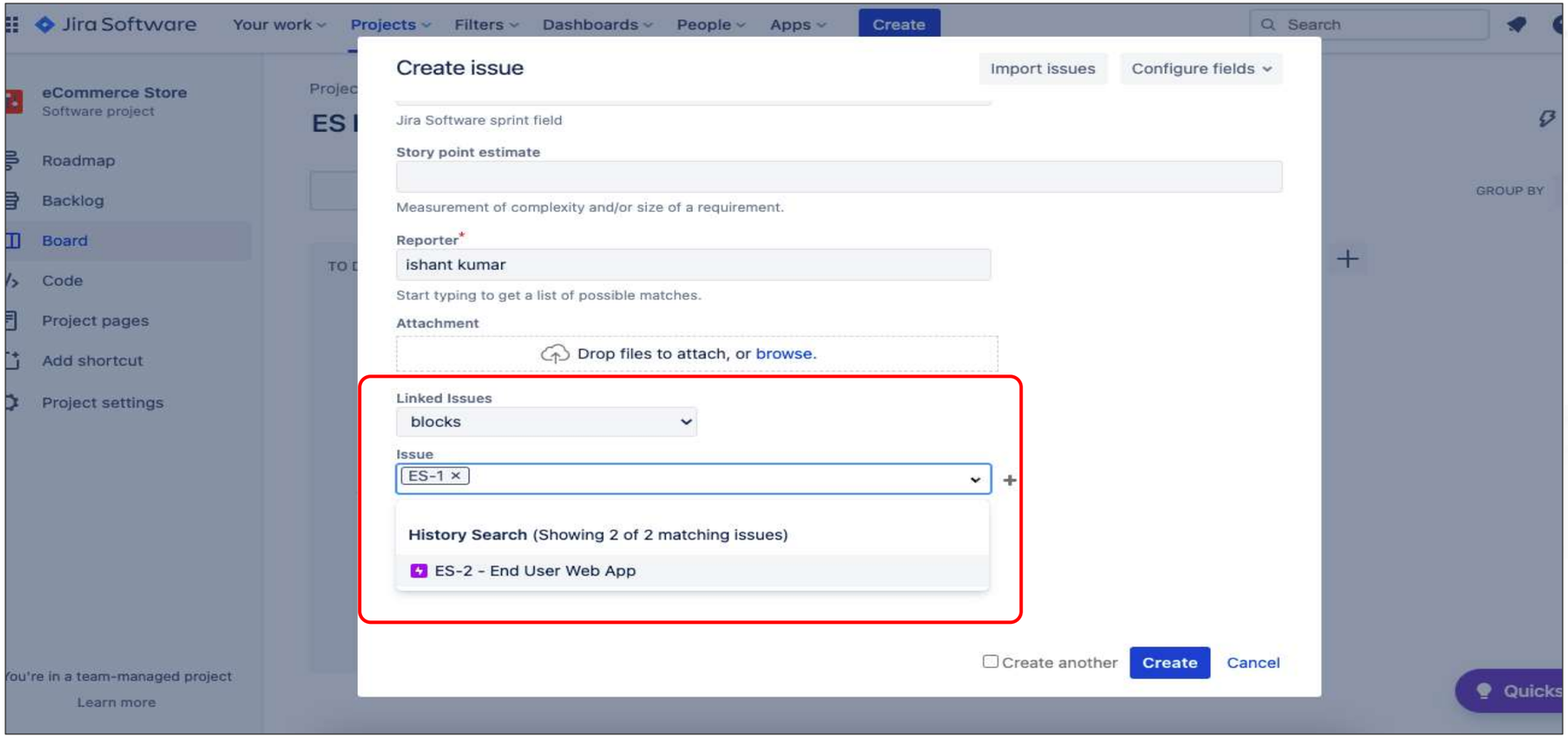
Authentication User Story

Similarly, we should create an issue type as **Story** for **Authentication** and provide details in the description section as shown below:



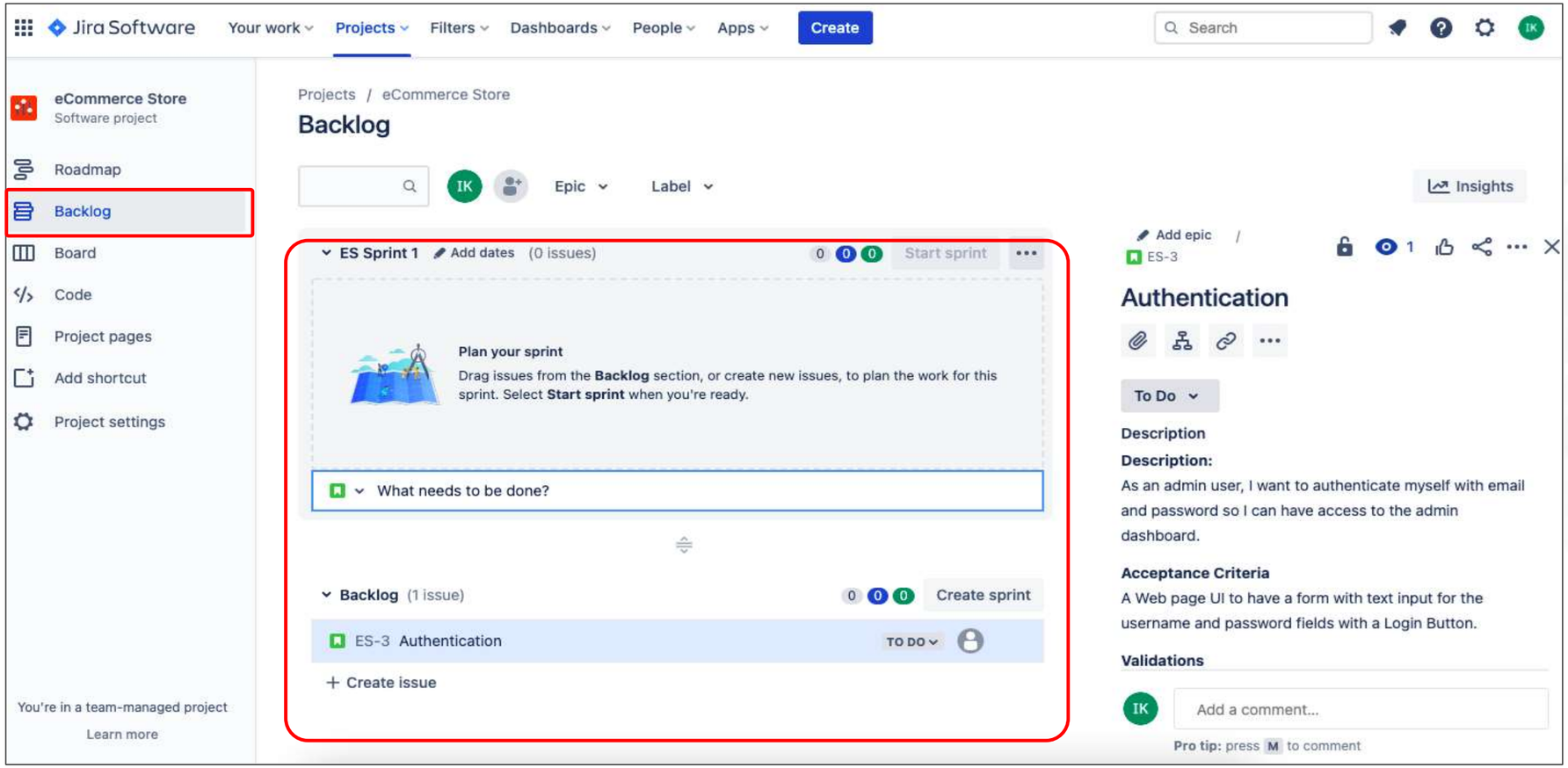
Link Story to Epic

Once the information is added, link the story to its corresponding Epic as shown below:



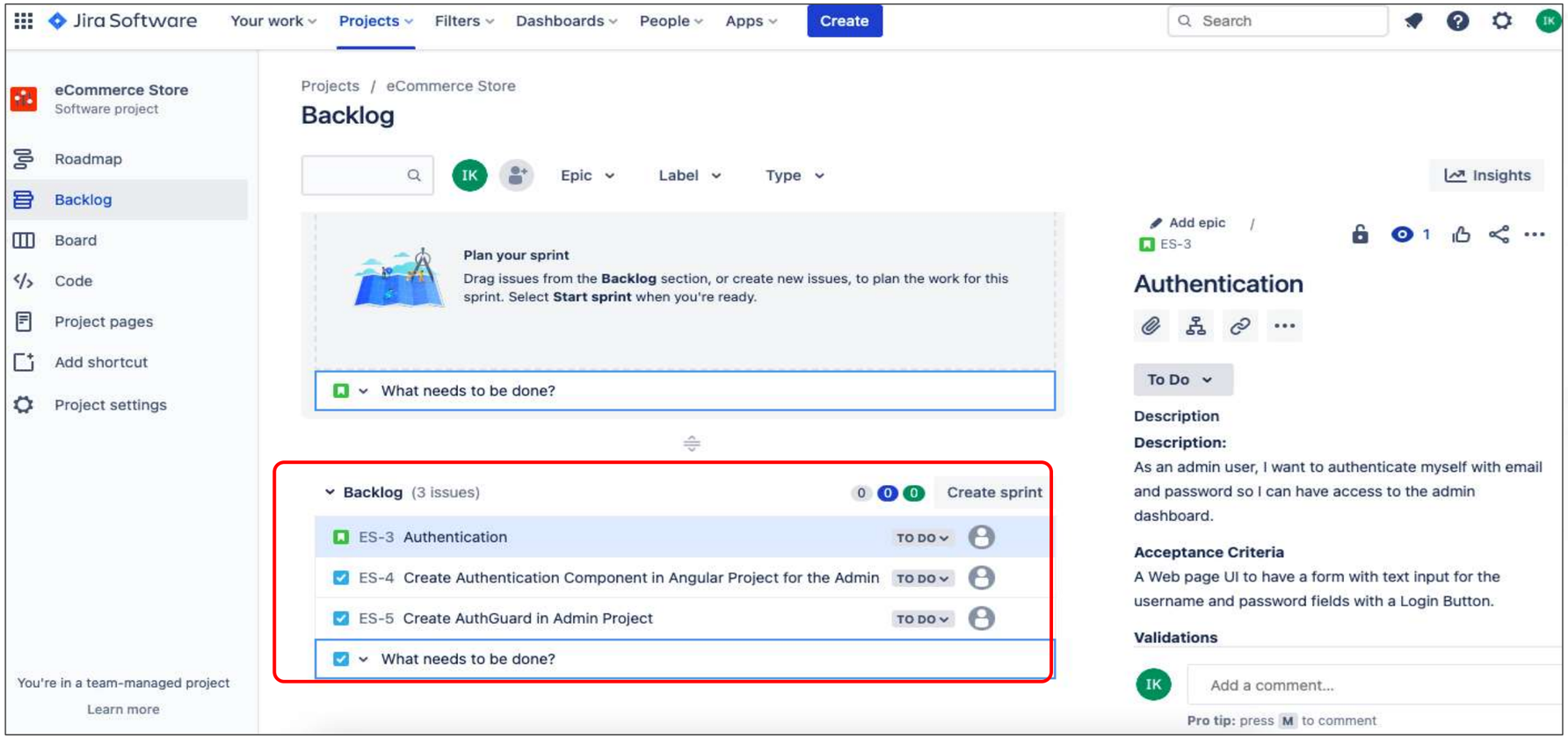
Backlog

After creating the Epic story, it will be listed in the Backlog on the JIRA Project Dashboard. Similarly create remaining Admin related user stories.



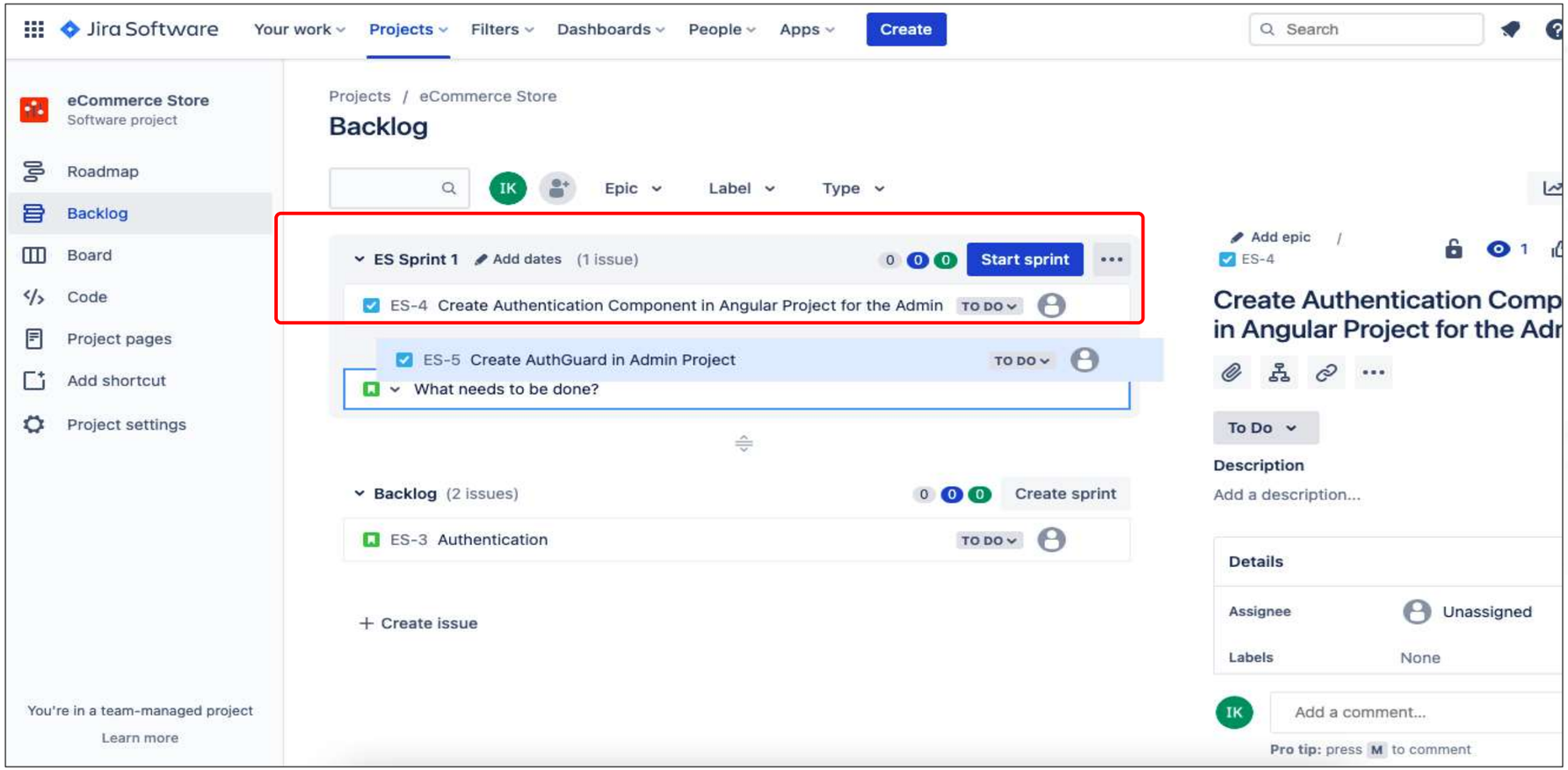
Tasks

Add the subtasks for the specific user story as shown below and perform the same process for remaining user stories



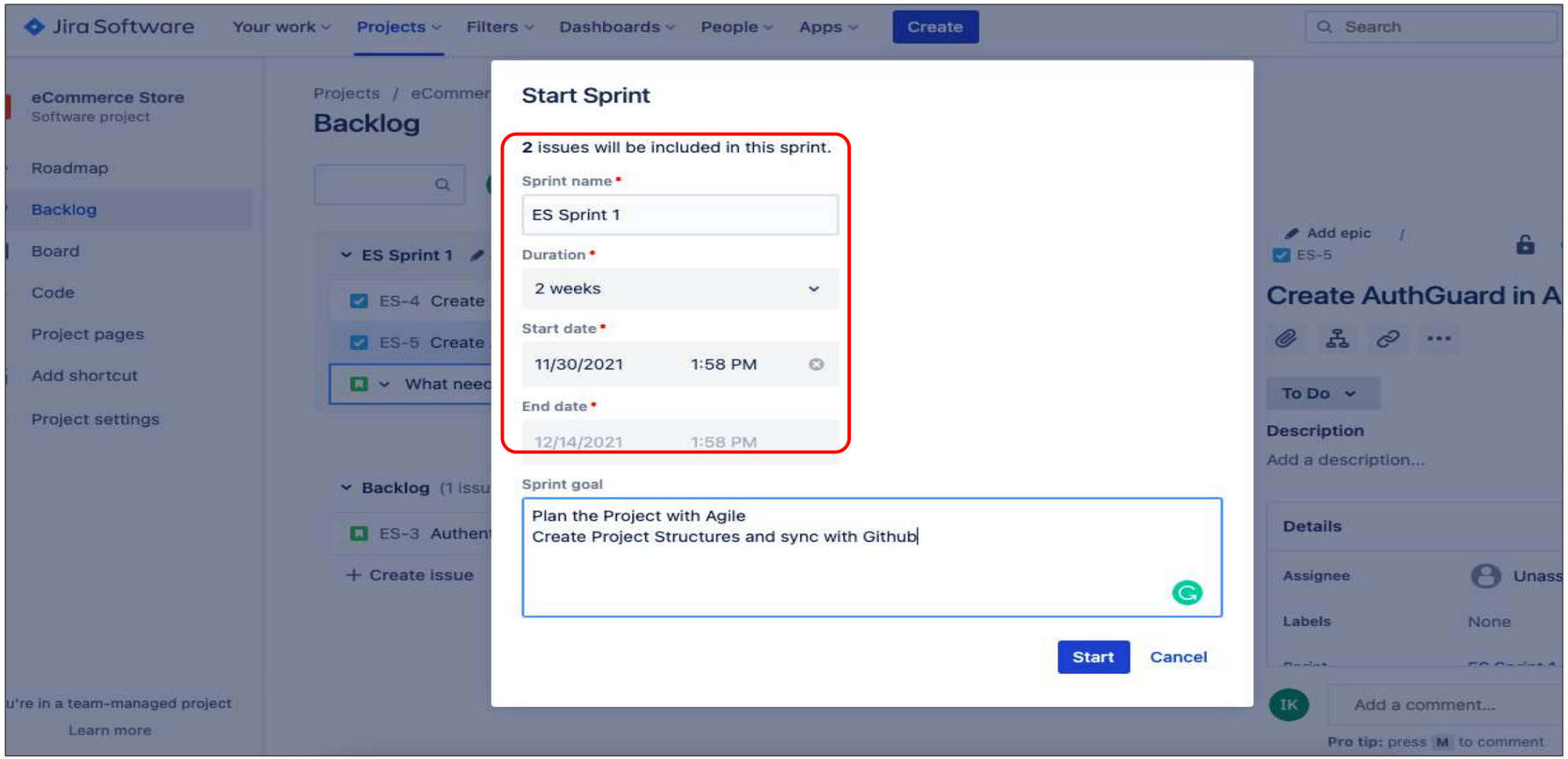
Sprint Planning

By default, a Sprint in the backlog section will be displayed by the name Sprint 1. Add the tasks in the sprint backlog based on the tasks that are planned for this session as mentioned in the scenario slide 13.



Starting a Sprint

To start the Sprint, fill in the details regarding name, duration, or customized date range for the duration.



User Stories: End-User Web App Epic

Authentication Story

The end user should be able to login into the end-user application. End user can access member features such as placing the order and checking order status.

The end user should be authenticated using email and password with an optional captcha verification. The register option should be made available for new users.

Authentication Story

A web page should have text fields for username and password with a login button for system access, along with another page for new user registration.

The username and password cannot be empty. Password must be of at least 6 characters.

Profile Story

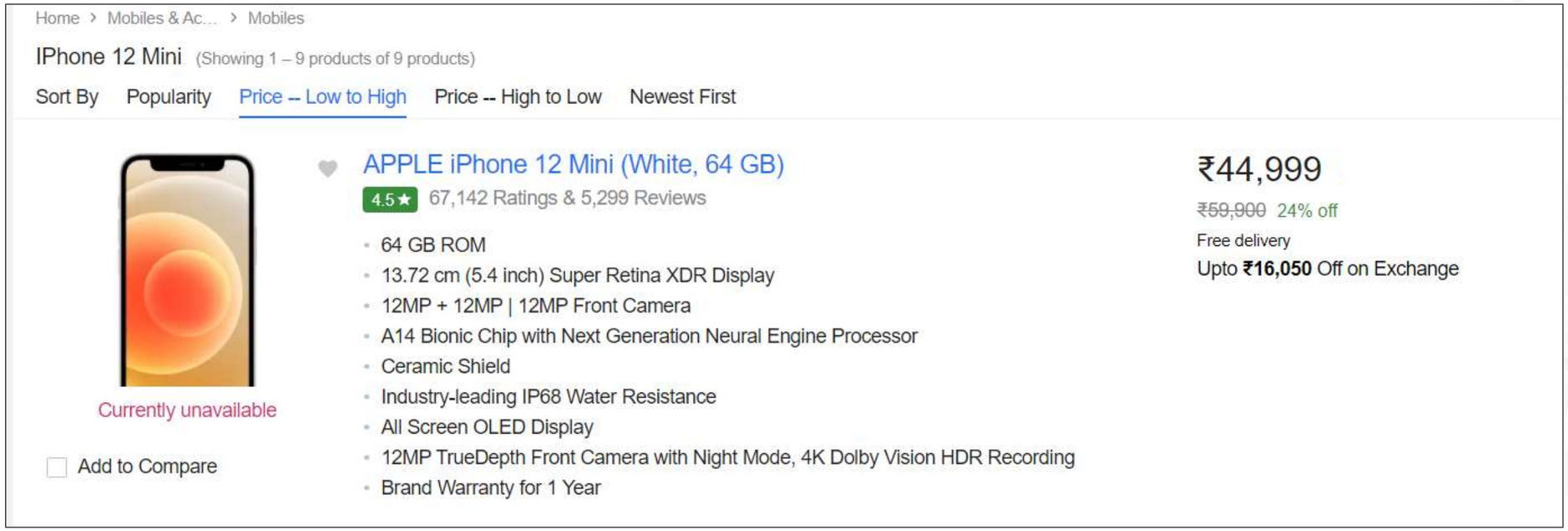
End user should be able to update basic information such as name, email, and delivery address.

A web page should have the option for the user to edit details.

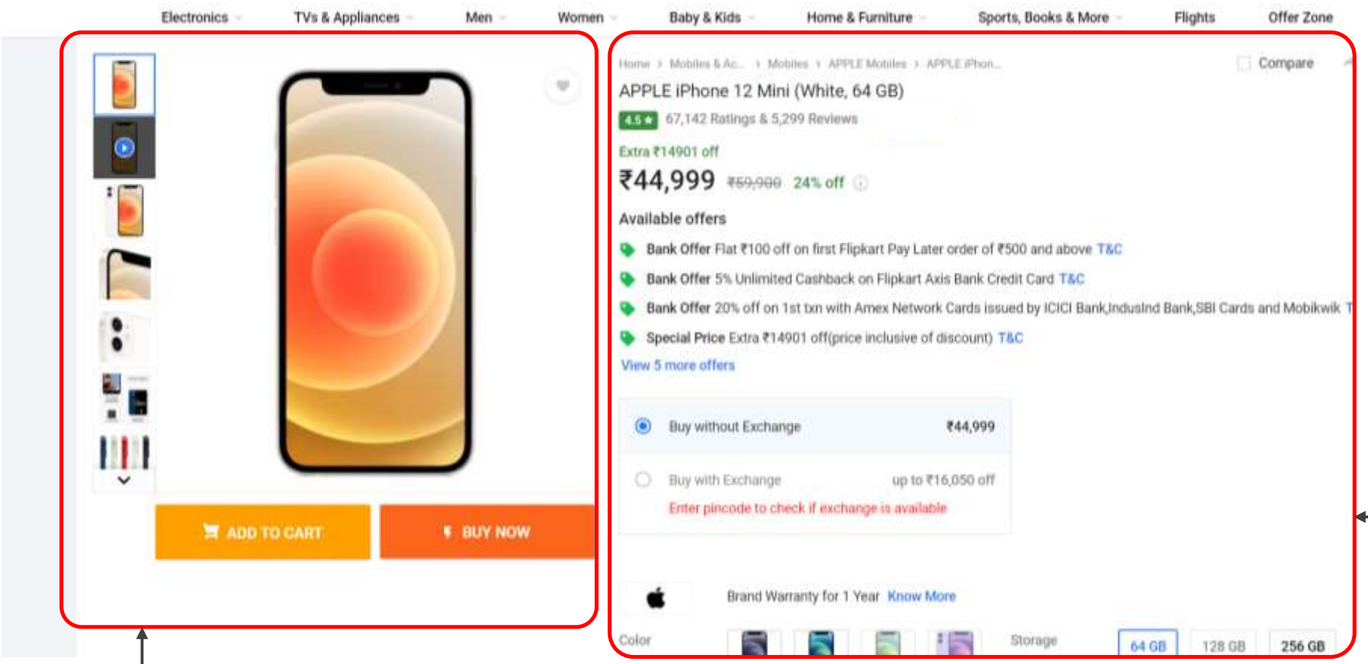
All the fields should be mandatorily filled.

Product Story

The end user should be able to view details of the product such as its image, material, user's review. Given below is an example of a sample product details:



Product Story

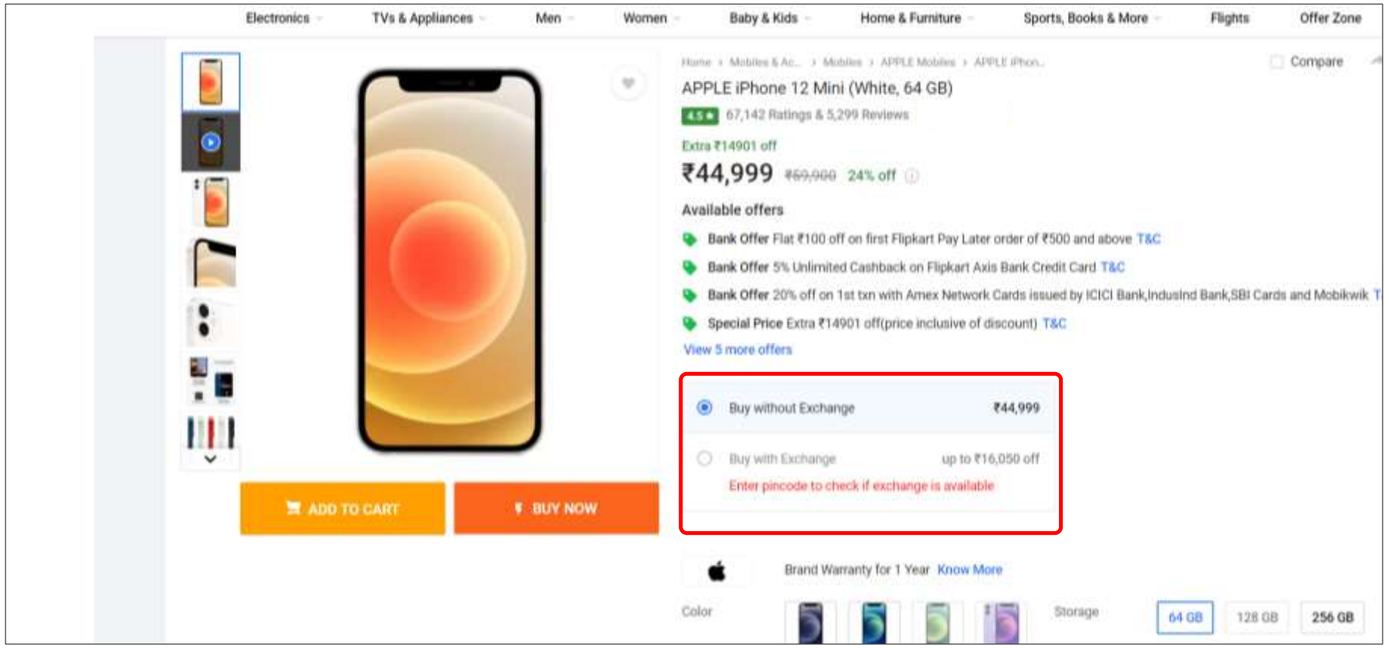


Product image and video section

Product description

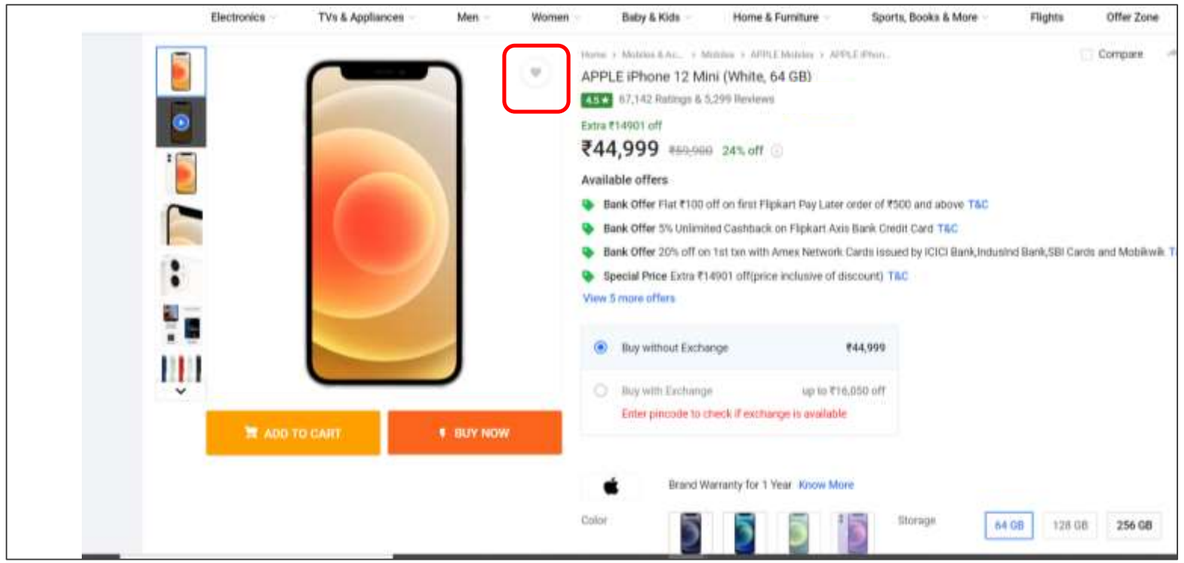
Product Story

Before placing the order, the availability of the product in the inventory should be validated based on the postal code of the user.



Wishlist Story

The end user should be able to add products to the Wishlist page.



Wishlist Story

There should be an option to add a product to the Wishlist for every product.

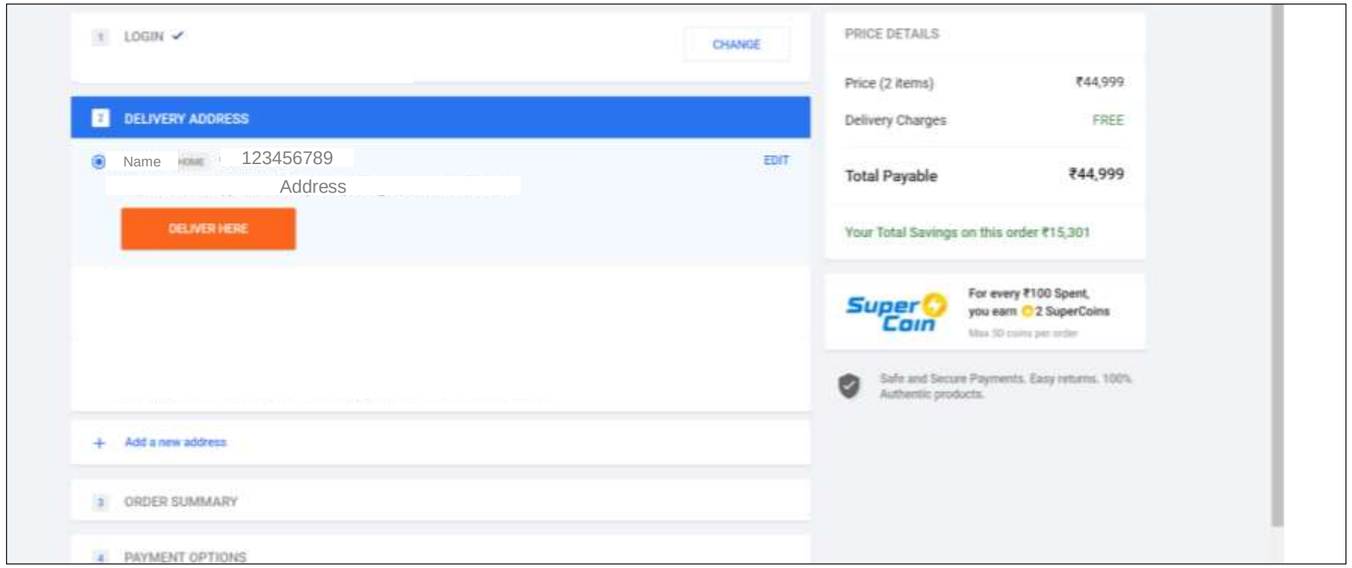
In the Wishlist page, a list of products should be displayed in the form of a list along with an option to place an order.

→

Availability of product in the inventory should be checked before adding products to the cart.

Checkout Story

The end user should be able to view the checkout page to order the products with payment options and delivery address.



The end user should be able to review the payment summary before placing an order. Once the order is placed, the end user should be able to check their order status.

End-user WebApp User Stories

Here is the list of the user stories for the End-user WebApp Epic:

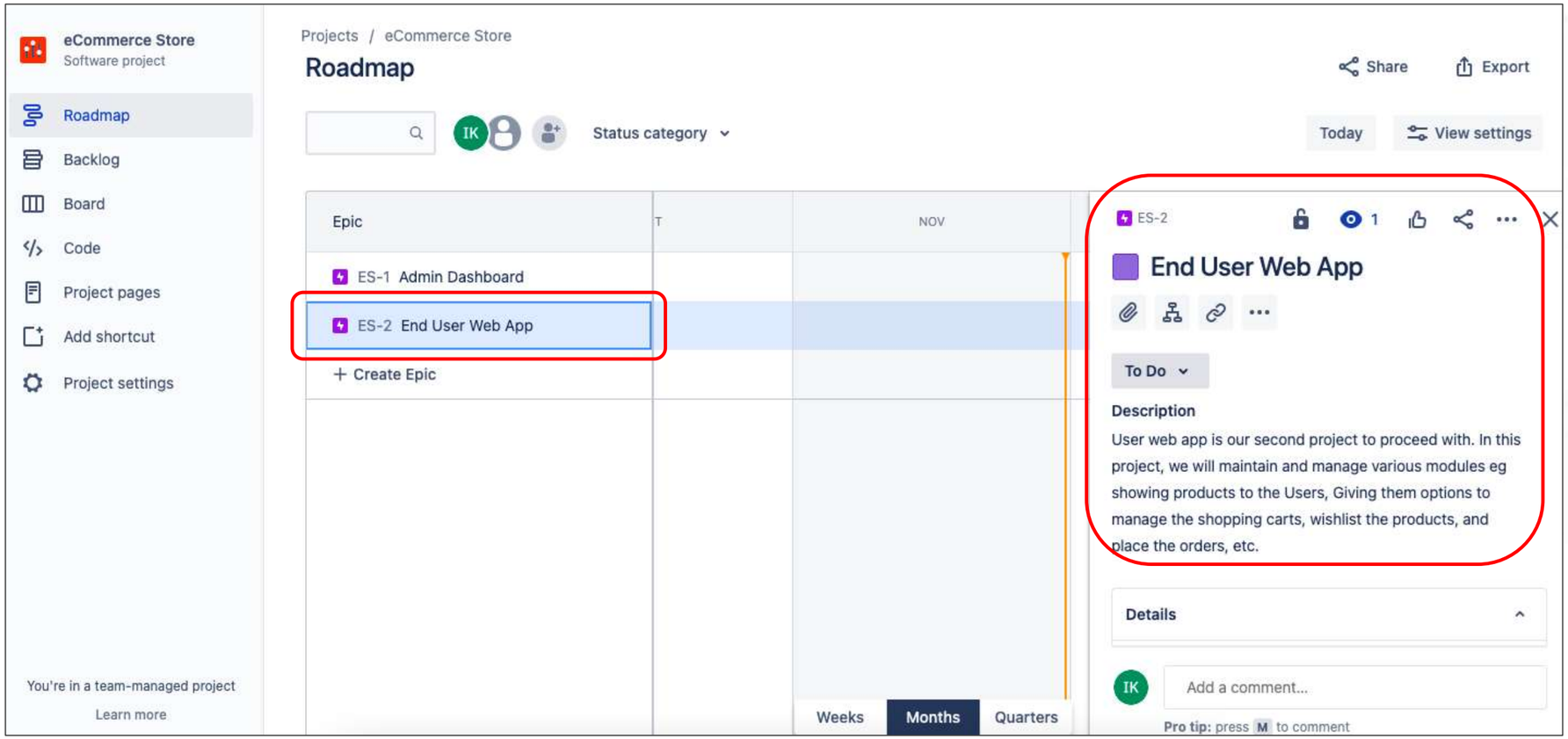
Role	User Story	Description	Acceptance Criteria
User	As a User, I want to create and manage my profile so that I can personalize my experience on the e-commerce platform.	Users can create, view, edit, and delete their profiles.	• User can create a profile with personal details like name, email, and address. • User can edit their profile information. • User can delete their profile.
User	As a User, I want to browse and search for products so that I can find items I am interested in purchasing.	Users can search for and view products in the catalog.	• User can use a search bar to find products by name or category. • User can filter and sort search results. • User can view detailed product information.
User	As a User, I want to add products to my Wishlist so that I can save items for future consideration.	Users can add and manage products in their Wishlist.	• User can add products to their Wishlist from the product page. • User can view and remove products from their Wishlist.

End-user Dashboard User Stories

Role	User Story	Description	Acceptance Criteria
User	As a User, I want to place orders for products so that I can purchase items from the e-commerce platform.	Users can add products to their cart and proceed to checkout.	• User can add products to their cart. • User can review their cart and proceed to checkout. • User can enter payment details and complete the purchase. • User receives an order confirmation email.
User	As a User, I want to view and track my orders so that I can see the status of my purchases.	Users can view their order history and track the status of current orders.	• User can see a list of their past orders. • User can view details and status of each order. • User can track the shipping status of their orders.
User	As a User, I want to manage my payment methods so that I can store and use different payment options.	Users can add, view, edit, and delete payment methods.	• User can add a new payment method. • User can edit existing payment methods. • User can delete a payment method.

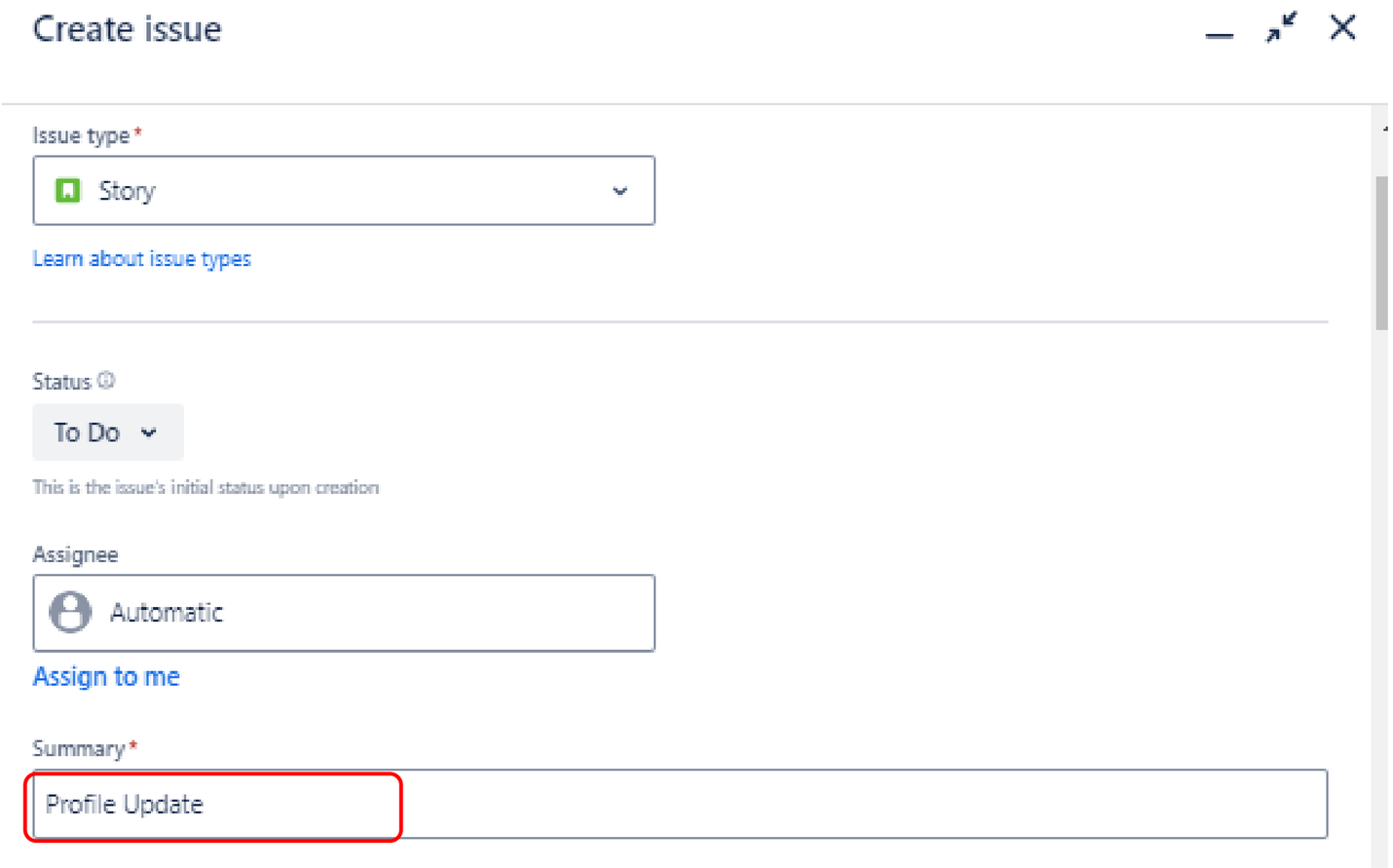
End-User Web App Epic

Added Epics will be viewable by the user in the Projects Roadmap. When selecting an epic, its summary and description should be visible.



End-User Profile Update Story

Create a story for Profile update or management for the end user as shown below:



End-User Profile Update Story

Add the description and acceptance criteria for the story. Similarly, make the remaining stories related to the end user.

Description

Normal text

Description:

As a User, I want to create and manage my profile so that I can personalize my experience on the e-commerce platform.

Acceptance Criteria:

- User can create a profile with personal details like name, email, and address.
- User can edit their profile information.
- User can delete their profile.

TECHNOLOGY

Creating Web Admin Dashboard Project Structure with Angular and Syncing with GitHub

simplilearn

Installing the Angular CLI

Install the Angular CLI to create a project with Angular.

The Angular CLI is a command–line interface tool used to initialize, develop, and maintain Angular applications through command shell.

Commands Used in Angular

Some useful commands are:

Task	Command
Generate a project	ng new my-first-project
Change directory in which project is created	cd my-first-project
Compile and execute the project	ng serve

Creating a New Angular Project

Create an Angular project using the following command in command shell.

```
ng new Admin-Dashboard
```

These options will appear:

```
? Would you like to add Angular routing? Yes
? Which stylesheet format would you like to use? CSS
```

- Enter *y* to enable routing and select CSS for styles.
- By default, git is initialized in the project to manage VCS.

Creating a New Angular Project

Following project structure should be created in a new Angular project:

```
CREATE admin-dashboard/README.md (1060 bytes)
CREATE admin-dashboard/.editorconfig (274 bytes)
CREATE admin-dashboard/.gitignore (604 bytes)
CREATE admin-dashboard/angular.json (3093 bytes)
CREATE admin-dashboard/package.json (1077 bytes)
CREATE admin-dashboard/tsconfig.json (783 bytes)
CREATE admin-dashboard/.browserslistrc (703 bytes)
CREATE admin-dashboard/karma.conf.js (1432 bytes)
CREATE admin-dashboard/tsconfig.app.json (287 bytes)
CREATE admin-dashboard/tsconfig.spec.json (333 bytes)
CREATE admin-dashboard/src/favicon.ico (948 bytes)
CREATE admin-dashboard/src/index.html (300 bytes)
CREATE admin-dashboard/src/main.ts (372 bytes)
CREATE admin-dashboard/src/polyfills.ts (2820 bytes)
CREATE admin-dashboard/src/styles.css (80 bytes)
CREATE admin-dashboard/src/test.ts (788 bytes)
CREATE admin-dashboard/src/assets/.gitkeep (0 bytes)
CREATE admin-dashboard/src/environments/environment.prod.ts (51 bytes)
CREATE admin-dashboard/src/environments/environment.ts (658 bytes)
CREATE admin-dashboard/src/app/app-routing.module.ts (245 bytes)
CREATE admin-dashboard/src/app/app.module.ts (393 bytes)
CREATE admin-dashboard/src/app/app.component.css (0 bytes)
CREATE admin-dashboard/src/app/app.component.html (24617 bytes)
CREATE admin-dashboard/src/app/app.component.spec.ts (1100 bytes)
CREATE admin-dashboard/src/app/app.component.ts (219 bytes)
```

Configuration Files Generated

Files	Use
.editorconfig	Configuration file for code editors
.gitignore	Files which git should ignore
README.md	A documentation for the introduction of application
angular.json	CLI configuration defaults for all projects in the workspace, including configuration options for build, serve, and test tools that the CLI uses
package.json	It configures node package manager dependencies that are available for all projects in the workspace.

Configuration Files Generated

Files	Use
package-lock.json	Provides version information for all packages installed into node_modules
src/	Is the directory where source files for the project reside
node_modules/	Is the directory where node_modules dependencies can be found
tsconfig.json	Is the typescript configuration for projects in the workspace

Web Admin Dashboard: Authentication Module

Authentication Module

Users who access the web admin dashboard, the authentication module is used for:

Validation of users

Authorization of users

Authentication Module

Components are the primary building block for Angular applications. Each component consists of:

An HTML template that declares what renders on the page

A Typescript class that defines behavior

A CSS selector that defines how the component is used in a template

Generating Auth Component

On terminal, change directory to user project root folder use the below command:

```
cd admin-dashboard
```

Then, execute the below command to generate auth component:

```
ng generate component auth
```

After execution, the following output is generated.

```
CREATE src/app/auth/auth.component.css
CREATE src/app/auth/auth.component.html
CREATE src/app/auth/auth.component.spec.ts
CREATE src/app/auth/auth.component.ts
UPDATE src/app/app.module.ts
```

Generating Auth Guard

AuthGuard is a class which implements the interface CanActivate.

- Generate auth guard in auth folder using command “ng generate guard auth/auth”, then select CanActivate interface to implement in this guard.

- After execution, the following output is generated.

```
CREATE src/app/auth/auth.guard.spec.ts
CREATE src/app/auth/auth.guard.ts
```

Generating Auth Service

For successful implementation, add Auth Service in the Project.

```
ng generate service auth/auth
```

After execution, the following output is generated.

```
CREATE src/app/auth/auth.service.spec.ts
CREATE src/app/auth/auth.service.ts
```

Web Admin Dashboard: Product Module

Generating Product-List Component

Product module includes product management. An admin user should implement the functionalities to manage products, such as:

- Pricing option
- Discount applicable
- Material description
- Size
- Quantity available
- Product images

Generating Product-List Component

Generate product-list component in products folder using the following command:

```
ng generate component products/productList
```

After execution, the following output is generated.

```
CREATE src/app/auth/product-list.component.css
CREATE src/app/auth/product-list.component.html
CREATE src/app/auth/product-list.component.spec.ts
CREATE src/app/auth/product-list.component.ts
UPDATE src/app/app.module.ts
```

Generating Product-View Component

To view the products, generate product-list component in products folder using the following command:

```
ng generate component products/productView
```

After execution, the following output is generated.

```
CREATE src/app/auth/product-view.component.css
CREATE src/app/auth/product-view.component.html
CREATE src/app/auth/product-view.component.spec.ts
CREATE src/app/auth/product-view.component.ts
UPDATE src/app/app.module.ts
```

TECHNOLOGY

Web Admin Dashboard: Users Module

simplilearn

Generating User Component

Generate user component using the following command:

```
ng generate component users
```

After execution, the following output is generated.

```
CREATE src/app/auth/users.component.css
CREATE src/app/auth/users.component.html
CREATE src/app/auth/users.component.spec.ts
CREATE src/app/auth/users.component.ts
UPDATE src/app/app.module.ts
```

Web Admin Dashboard: Orders Module

Generating Orders Component

The admin dashboard must have insights on the orders placed by end users from their web app. Few of the details which would be required include:

- View order details which will include products and their quantity.
- Check the availability of products in order, and if not available, notify the user accordingly either manually or by the algorithmic approach.

Generating Orders Component

Generate orders component using the following command:

```
ng generate component orders
```

After execution, the following output is generated.

```
CREATE src/app/auth/orders.component.css
CREATE src/app/auth/orders.component.html
CREATE src/app/auth/orders.component.spec.ts
CREATE src/app/auth/orders.component.ts
UPDATE src/app/app.module.ts
```

Web Admin Dashboard: Payment Module

Generating Payment Component

Admin must have the data for the transactions associated with the order, when user places it.

An admin must be able to:

- Manage payments
- Reject payments
- Process payments partially

Generating Payment Component

Generate user's payment using the below command:

```
ng generate component payment
```

After execution, the following output is generated.

```
CREATE src/app/auth/payment.component.css
CREATE src/app/auth/payment.component.html
CREATE src/app/auth/payment.component.spec.ts
CREATE src/app/auth/payment.component.ts
UPDATE src/app/app.module.ts
```

Generating Payment Service

Services in Angular increase modularity and reusability.

Generate payment service in payment folder using this command;

```
ng generate service payment/payment
```

After execution, the following output is generated.

```
CREATE src/app/auth/payment.service.spec.ts
CREATE src/app/auth/payment.service.ts
```

Web Admin Dashboard: Shipment Module

Generating Shipment Component

In the shipment module of admin dashboard, the list of orders ready for shipment should be shown.

- It should implement management of order transition from warehouse to user's doorstep.
- It should update order details and notify the customer.

Generating Shipment Component

Generate shipment component using the below command:

```
ng generate component shipment
```

After execution, the following output is generated.

```
CREATE src/app/auth/shipment.component.css
CREATE src/app/auth/shipment.component.html
CREATE src/app/auth/shipment.component.spec.ts
CREATE src/app/auth/shipment.component.ts
UPDATE src/app/app.module.ts
```

Building the Project

To build the project, execute:

```
npm run build
```

To test the output on the localhost, use:

```
> ng serve -o
```

After execution, the following output is generated.

```
✓ Browser application bundle generation complete.
✓ Copying assets complete.
✓ Index html generation complete.

Initial Chunk Files                    | Names          |       Size
main.bb310dfd8e03708854b9.js           | main           | 214.81 kB
polyfills.9f1d9ffccfb9cf2e763b.js      | polyfills      |  36.21 kB
runtime.f3142626aa6e5ec05e95.js        | runtime        |   1.03 kB
styles.31d6cfe0d16ae931b73c.css        | styles         |   0 bytes
```

Web Admin Dashboard: Pushing Project on GitHub

GitHub

Git is a distributed version control system designed to efficiently handle small to big projects.

In this project, Git is used for VCS and syncing projects in GitHub.

Git Commands

Type the command to add all the files

```
git add .
```

Check git status for the files added

```
git status
Output of git status:
```

Git Command

Commit the files added for push operation

```
git commit -m "initial commit"
Output of git commit:
[master 42e3828] initial commit
 Committer: username <admin@username-MacBook-Air.local>
```

Git Remote Command

Git remote commands are used for setting remote to track local repository.

To add remote repository, use the below command:

```
git remote add origin https://github.com/github-username/admin-dashboard.git
```

Git Push Command

Git push command is used to push local changes to the GitHub repository from command line.

Use the below command:

```
git push origin master
```

Creating Web App Project Structure for End Users with Angular and Syncing with GitHub

Web App Project Structure

User web app is the second project to proceed. In this project, various modules should be managed, such as:

- Showing products to the users
- Giving users options to manage the shopping carts, such as Wishlist the products and place the orders.

Creating a New Angular Project

Create the admin dashboard by executing the below command on your terminal or shell.

```
ng new user-web-app
```

Following options will appear:

```
? Would you like to add Angular routing? Yes
? Which stylesheet format would you like to use? CSS
```

Creating a New Angular Project

As a result, the project structure shown below will be created.

```
CREATE user-web-app/README.md (1060 bytes)
CREATE user-web-app/.editorconfig (274 bytes)
CREATE user-web-app/.gitignore (604 bytes)
CREATE user-web-app/angular.json (3093 bytes)
CREATE user-web-app/package.json (1077 bytes)
CREATE user-web-app/tsconfig.json (783 bytes)
CREATE user-web-app/.browserslistrc (703 bytes)
CREATE user-web-app/karma.conf.js (1432 bytes)
CREATE user-web-app/tsconfig.app.json (287 bytes)
CREATE user-web-app/tsconfig.spec.json (333 bytes)
CREATE user-web-app/src/favicon.ico (948 bytes)
CREATE user-web-app/src/index.html (300 bytes)
CREATE user-web-app/src/main.ts (372 bytes)
CREATE user-web-app/src/polyfills.ts (2820 bytes)
CREATE user-web-app/src/styles.css (80 bytes)
CREATE user-web-app/src/test.ts (788 bytes)
CREATE user-web-app/src/assets/.gitkeep (0 bytes)
CREATE user-web-app/src/environments/environment.prod.ts (51 bytes)
CREATE user-web-app/src/environments/environment.ts (658 bytes)
CREATE user-web-app/src/app/app-routing.module.ts (245 bytes)
CREATE user-web-app/src/app/app.module.ts (393 bytes)
CREATE user-web-app/src/app/app.component.css (0 bytes)
CREATE user-web-app/src/app/app.component.html (24617 bytes)
CREATE user-web-app/src/app/app.component.spec.ts (1100 bytes)
CREATE user-web-app/src/app/app.component.ts (219 bytes)

Lastly you will see,
✓ Packages installed successfully.
    Successfully initialized git.
```

Web App for End User: Authentication Module

Generating Auth Component: Login

Open project in the terminal, then use the below command:

```
ng generate component auth/login
```

After execution, the following output is generated.

```
CREATE src/app/auth/login/login.component.css
CREATE src/app/auth/login/login.component.html
CREATE src/app/auth/login/login.component.spec.ts
CREATE src/app/auth/login/login.component.ts
UPDATE src/app/app.module.ts
```

Generating Auth Component: Register

Open project in the terminal, then use the below command:

```
ng generate component auth/register
```

After execution, the following output is generated.

```
CREATE src/app/auth/register/register.component.css
CREATE src/app/auth/register/register.component.html
CREATE src/app/auth/register/register.component.spec.ts
CREATE src/app/auth/register/register.component.ts
UPDATE src/app/app.module.ts
```

Generating Auth Guard

Open project in the terminal, then use the command:

```
ng generate guard guards/auth
```

After execution, the following output is generated.

```
CREATE src/app/guards/auth.guard.spec.ts
CREATE src/app/guards/auth.guard.ts
```

Generating Auth Service

Open project in the terminal, then use the below command:

```
ng generate service services/auth
```

After execution, the following output is generated.

```
CREATE src/app/services/auth.service.spec.ts
CREATE src/app/services/auth.service.ts
```

Web App for End User: Profile Module

Generating Profile Component

User profile should contain the essential details of the user who has registered on the web app.

It includes basic information such as:

- Name, email, phone number, and profile image (optional)
- List of delivery addresses

Generating Profile Component

Open project in the terminal, then use the below command:

```
ng generate component pages/profile
```

After execution, the following output is generated.

```
CREATE src/app/pages/profile/profile.component.css
CREATE src/app/pages/profile/profile.component.html
CREATE src/app/pages/profile/profile.component.spec.ts
CREATE src/app/pages/profile/profile.component.ts
UPDATE src/app/app.module.ts
```

Web App for End User: Product Module

Generating Product Component

- Product component is used to list all the products to the users to buy or add to their shopping cart or Wishlist for future purchase.
- It should include the product listing.
- The product list should be shown to the end user with the below details:

Pricing Option	Discount Applicable	Material Description
Size	Quantity Available	Product Images

Generating Product Component

Open project in the terminal, then use the below command:

```
ng generate component pages/products
```

After execution, the following output is generated.

```
CREATE src/app/pages/products/products.component.css
CREATE src/app/pages/products/products.component.html
CREATE src/app/pages/products/products.component.spec.ts
CREATE src/app/pages/products/products.component.ts
UPDATE src/app/app.module.ts
```

Web App for End User: Shopping Cart and Shipping Module

Generating Shopping Cart Component

The shopping cart component should show the user the products added to the cart for making the purchase.

- Checkout option should allow the user to review the final details of the order and then proceed with payments.
- A payment method from the list of payment methods should be available to place an order at the checkout time.

Generating Shopping Cart Component

Open project in the terminal, then use the below command:

```
ng generate component pages/shoppingCart
```

After execution, the following output is generated.

```
CREATE src/app/pages/shopping-cart/shopping-cart.component.css
CREATE src/app/pages/shopping-cart/shopping-cart.component.html
CREATE src/app/pages/shopping-cart/shopping-cart.component.spec.ts
CREATE src/app/pages/shopping-cart/shopping-cart.component.ts
UPDATE src/app/app.module.ts
```

Generating Checkout Component

Open project in the terminal, then use the below command;

```
ng generate component pages/checkout
```

After execution, the following output is generated.

```
CREATE src/app/pages/checkout/checkout.component.css
CREATE src/app/pages/checkout/checkout.component.html
CREATE src/app/pages/checkout/checkout.component.spec.ts
CREATE src/app/pages/checkout/checkout.component.ts
UPDATE src/app/app.module.ts
```

Building the Project

To build the project, execute;

```
npm run build
```

To test the output on the localhost, use;

```
> ng serve -o
```

After execution, the following output is generated.

```
✓ Browser application bundle generation complete.
✓ Copying assets complete.
✓ Index html generation complete.

Initial Chunk Files                          | Names          |       Size
main.bb310dfd8e03708854b9.js                 | main           | 214.81 kB
polyfills.9f1d9ffccfb9cf2e763b.js            | polyfills      |  36.21 kB
runtime.f3142626aa6e5ec05e95.js              | runtime        |   1.03 kB
styles.31d6cfe0d16ae931b73c.css              | styles         |   0 bytes
```

Web App for End User: Pushing Project on GitHub

Git Command

Git is initialized by default when angular project is created.

Type the command to add all the files:

```
git add .
```

Check git status for the files added.

```
git status
Output of git status:
```

Git Command

Commit the files we added for push operation

```
git commit -m "initial commit"
Output of git commit:
[master 42e3828] initial commit
 Committer: username <admin@username-MacBook-Air.local>
```

Git Remote Command

Git remote commands are used for setting remote for tracking local repository.

To add remote repository, use the below command:

```
git remote add origin https://github.com/github-username/user-web-app.git
```

Git Push Command

Finally, Git push command is used to push local changes to the GitHub repository from command line.

Use the below command:

```
git push origin master
```

Key Takeaways

- As an admin user, he should perform the following activities on the web page, such as:
    - Adding product detail
    - Accessing user's data
    - Monitoring newly registered users
    - Monitoring frequently visited sections by the users
    - Monitoring and viewing the list of orders
    - Checking availability of order in inventory
    - Monitoring payment status of the order

Key Takeaways

- As an end-user, he should perform the following activities on the web page, such as:
  - Checking the product details
  - Checking availability of the product
  - Adding products to the Wishlist
  - Placing order Reviewing payment summary before placing an order
- JIRA is used to create a project to manage with Agile.
- Scrum project theme in JIRA is useful to plan and create Sprints and break down a complex project into small projects.

Key Takeaways

- The Angular CLI is a command-line interface tool used to initialize, develop, and maintain Angular applications through a command shell.
- Components are the primary building blocks for Angular applications. Each component consists of an HTML template, a Typescript class, a CSS selector, and optionally CSS style.
- Git is initialized by default when the angular project is created and is used for VCS and syncing projects in GitHub.