Caltech | Center for Technology & Management Education

Live Session 01: Project Planning and Create Project Structure

Powered by simplilearn

Caltech | Center for Technology & Management Education

Project Planning

Powered by simplilearn

You Already Know

Before we begin, let's recall what we have covered till now:

- Agile 
- Git 
- MySQL 
- HTML 
- CSS 
- JavaScript 

Recap

Agile

It is an iterative approach used to manage the development of a software project.

Git

It is a distributed version control system used to handle software projects.

MySQL

It is a Relational Database Management System used to store data in a structured manner using tables.

Recap

HTML and CSS

It is a designed interactive web page.

JavaScript

It is a programming language used in web pages.

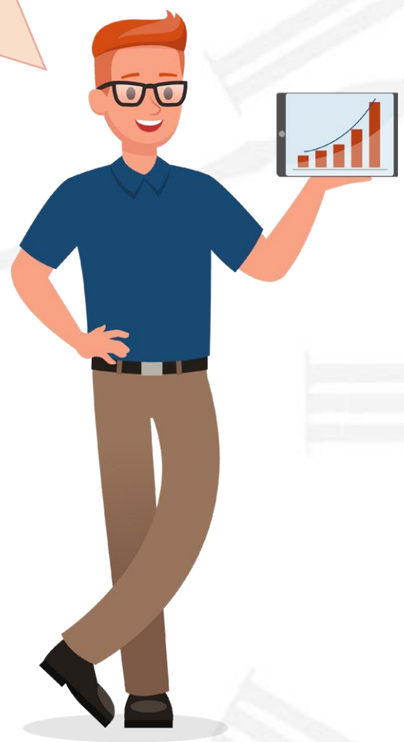
A Day in the Life of a Full Stack Developer

Meet Mr. Bob. He is an associate developer working for ZeeCrop.

A Day in the Life of a Full Stack Developer

Meet Mr. Mark. He is a product owner of the team Bob is a part of.

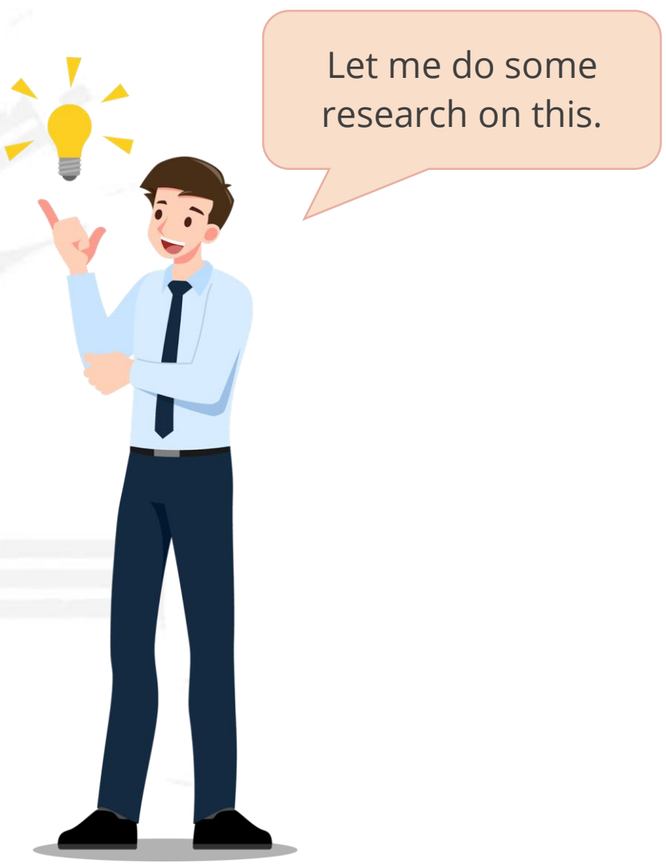
Hi Bob! I want to increase our product sales. For that, I require a user-friendly web page. Users should be able to find all their requirements on a single page. Can you build that kind of web page for us?



Sure, Mark. I will do that for you.

A Day in the Life of a Full Stack Developer

Now, Mr. Bob thinks about Mr. Mark's requirements and is puzzled about designing a web page where all the options are available on a single page.



A Day in the Life of a Full Stack Developer

He finds the following helpful information to build his web page through his research.

- Angular and Node can be used to build front end of the web page.
- Spring Boot, Java, and MySQL/MongoDB can be used to build at the back end.
- Agile methodology is best for project planning.
- Docker and cloud technologies are used to deploy the application.

A Day in the Life of a Full Stack Developer

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

Bob decides to do following activities:

- Create user stories for Epics of Web App for 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:

- Explain the various aspects in developing the user stories
- Develop user stories for admin dashboard epic and web app for end user epic
- Execute stories and plan sprints in JIRA

Learning Objectives

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

- Execute command using the Angular CLI
- Create a Web App Project for 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 for Web Admin Dashboard Epic

Admin Dashboard

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

Authentication Story

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

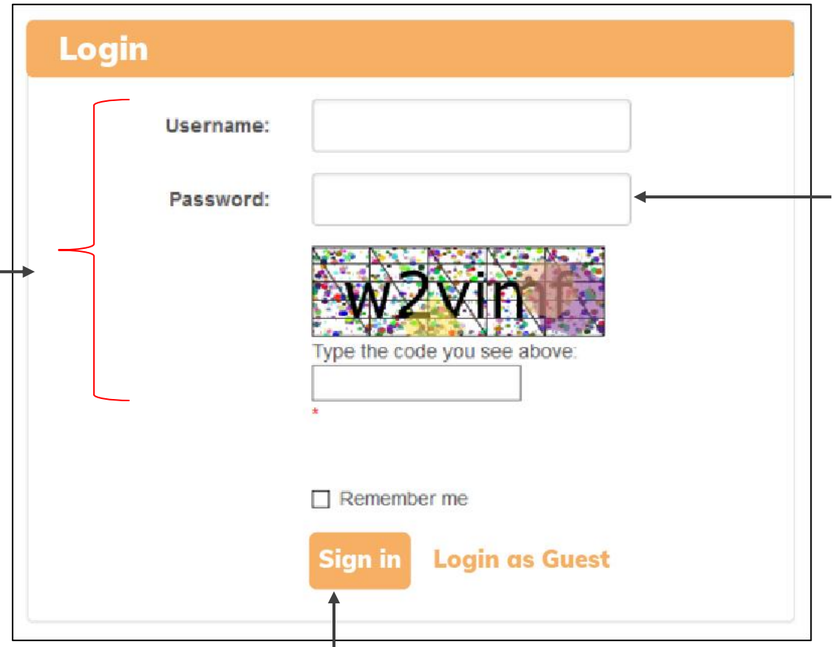
Admin user should perform identification authentication using email and password with captcha verification to access admin site at the backend.

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 6 characters minimum.

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



Password should be a minimum of 6 characters.

NFR: User should be navigated to the dashboard page in less than 3 seconds after logging in.

Product Story, Product List, and Product Details

Admin user should be able to manage products and add details regarding product, such as:

Price
and Discount

Material and
Size Description

Availability of
Quantity

Product Usage
Video Tutorial

Product Story, Product List, and Product Details

The screen should be divided into two parts. The first part should display the list of products, and the second part should show the description of the product.

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:

User's Story

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

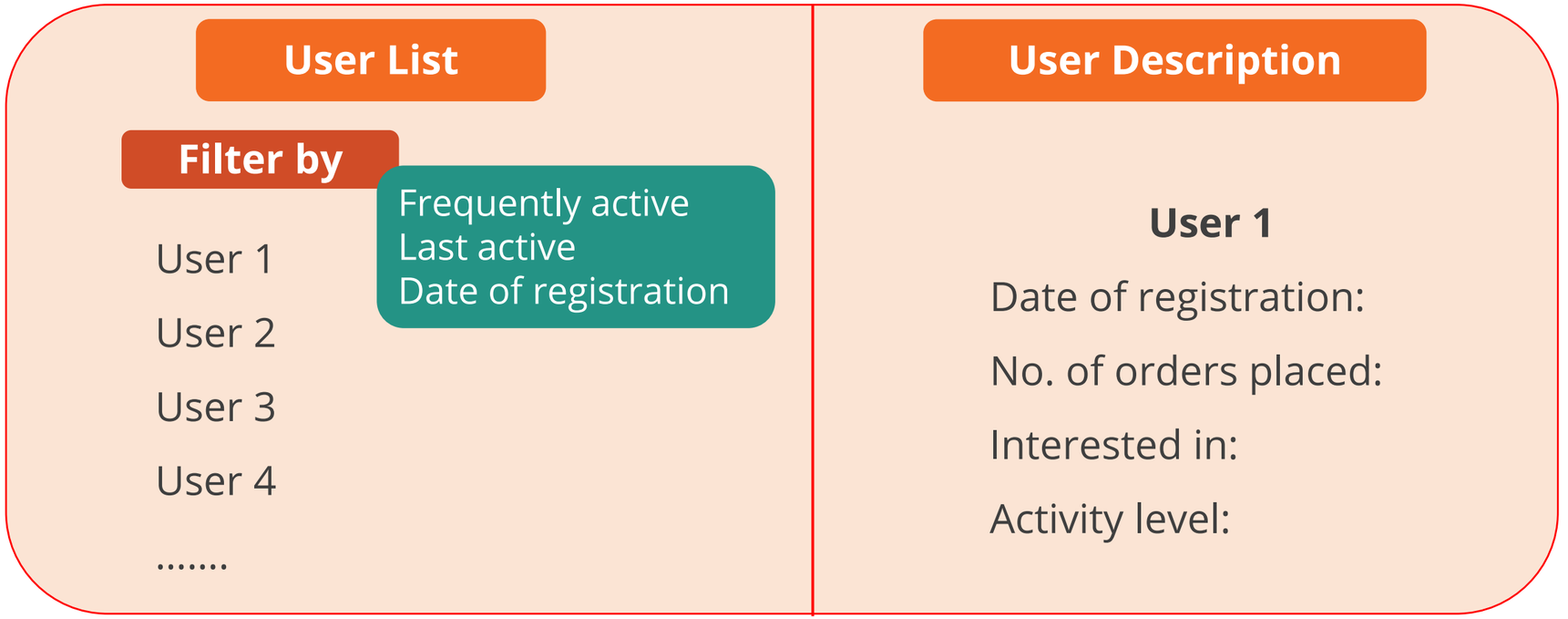
User’s Story

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.

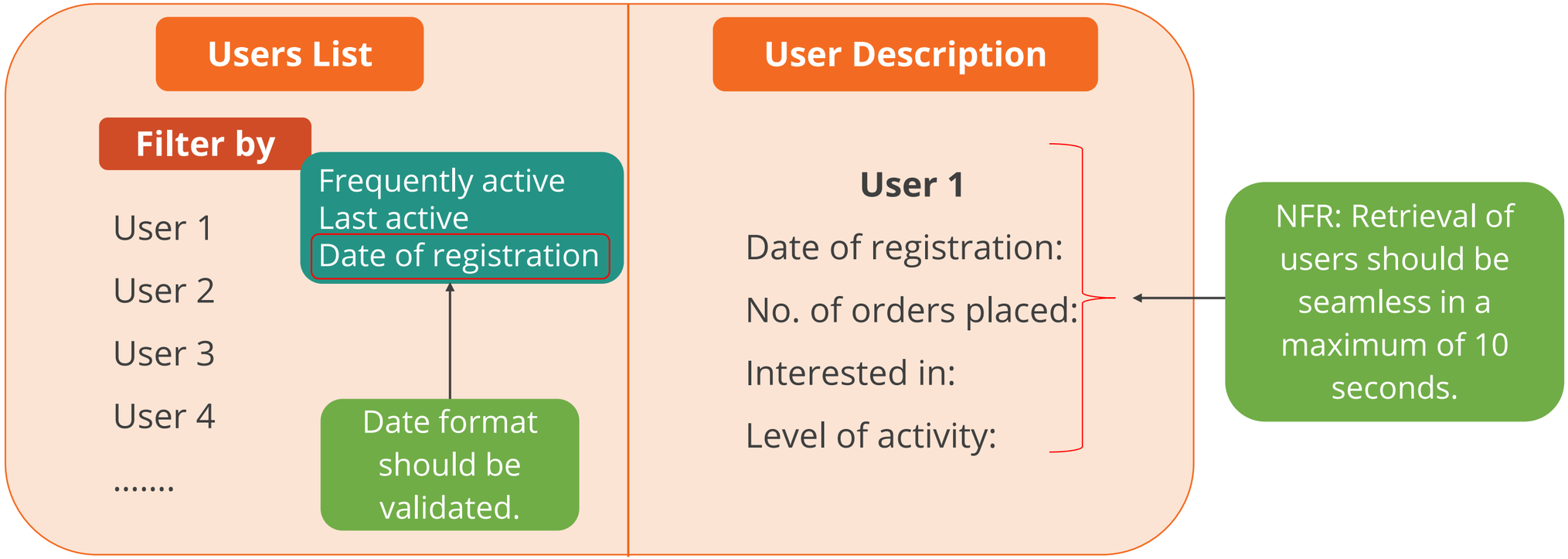
The first part shows a list of users, and the second part shows user descriptions, such as date of registration and activity level.

User's Story


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:

User's Story

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



Order Story

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

Admin user 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.

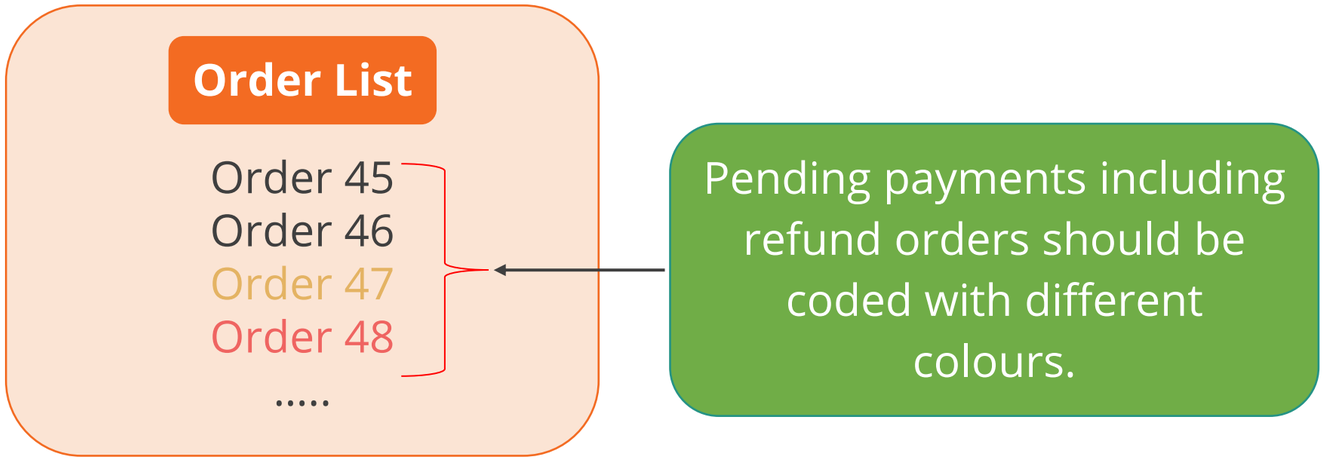
New Orders	Ongoing Orders	Halt State
Order 45	Order 39	Order 28
Order 46	Order 40	Order 30
Order 47	Order 43	Order 35
Order 48	Order 44	Order 41
.....	.....	.....

Order Story

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

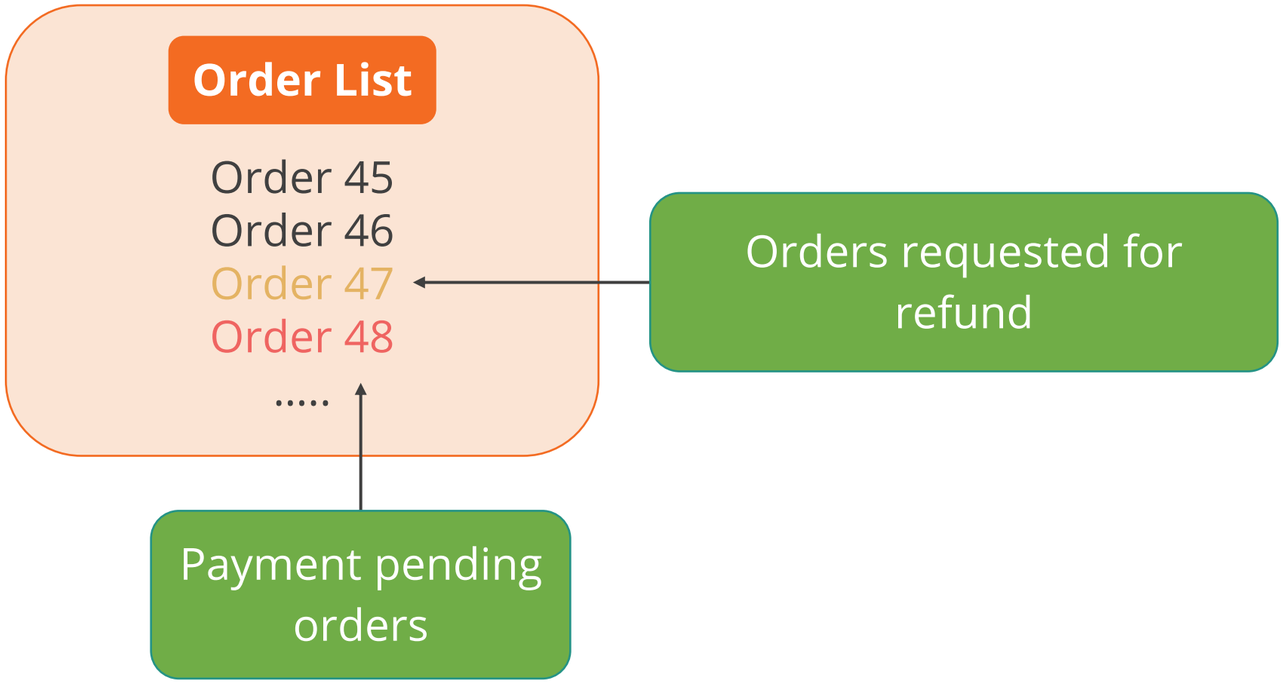
Payment Story

Admin user should be able to monitor and view the list of orders whose payment status is pending or in a disputed state.



Payment Story

On the web page, a list of all pending payments, including orders requested for refund should be color coded to differentiate for easy recognition.



Shipping Story

Admin user should be able to view orders which are ready for shipment and in transition.

Admin user should be able to update the delivery date, package's current location for end user reference.

User Stories for End User Web App Epic

Authentication Story

The end user should be able to login into the end-user app. Also, acquire access member features such as placing the order and checking order status.

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

Authentication Story

A web page with text fields for username and password with a login button to log in to the system should be available, along with another web page for new user registration.

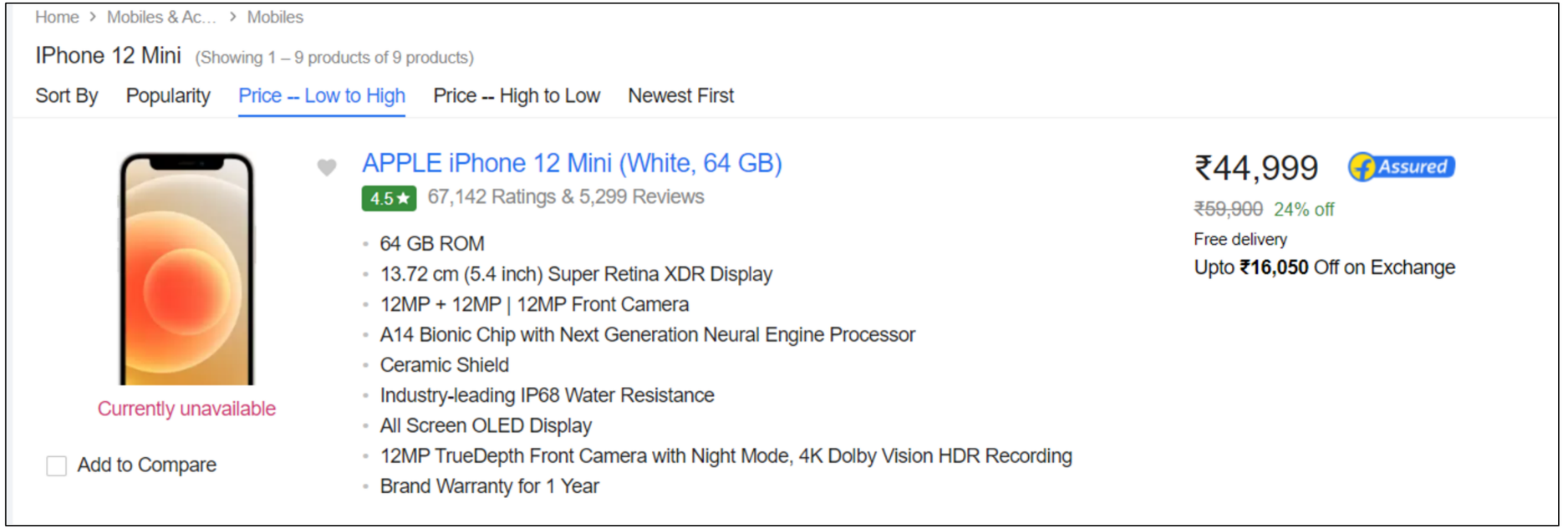
The username and password cannot be empty. Password must be of 6 characters minimum.

Profile Story

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

Product Story

The end user should view details of the product such as its image, material, user's review.

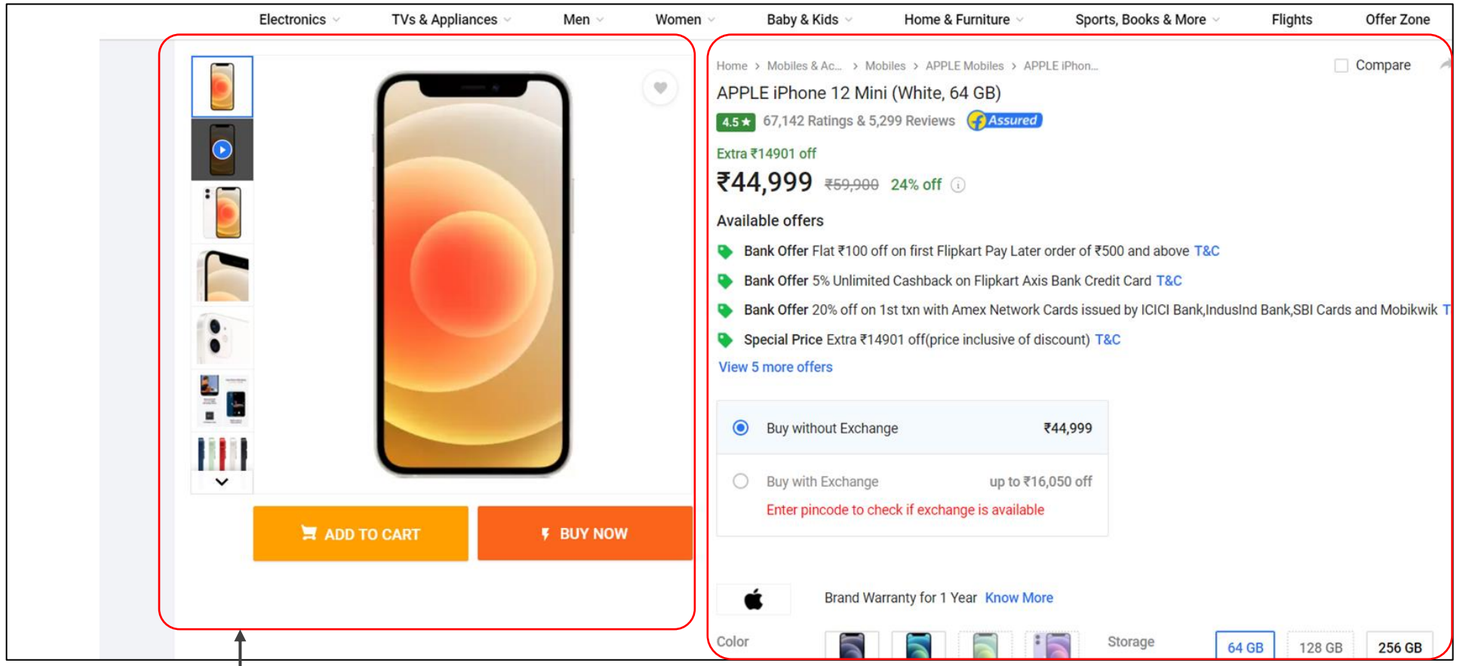


Product Story

A web page should have two sections.

- The first section should show product images and video.
- The second section should show product pricing, material description.
- User reviews should be under both sections, with the latest ones on the top of the list.

Product Story

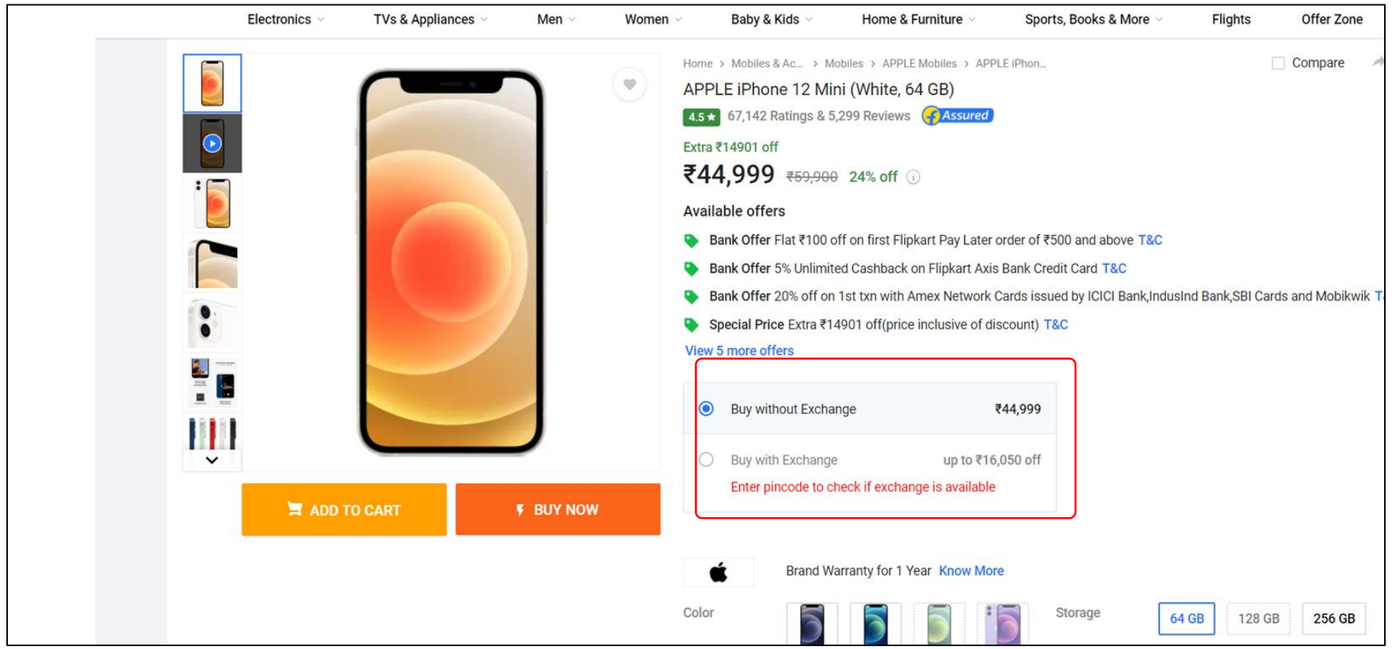


Product image and video section

Product description

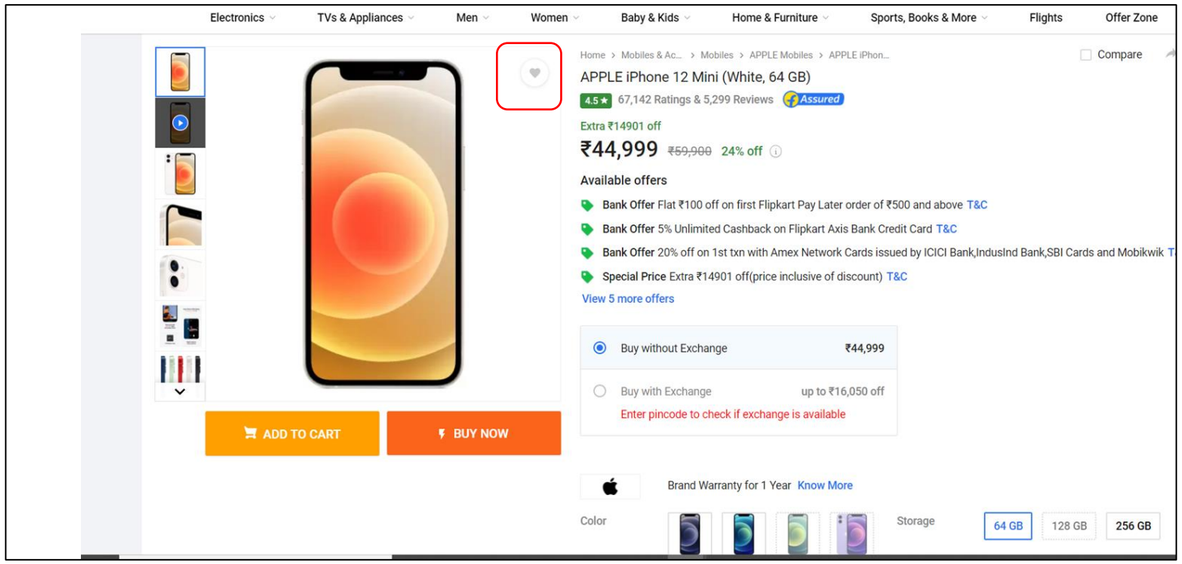
Product Story

Before placing the order, the availability of the product in the inventory should be validated.



Wishlist Story

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



Wishlist Story

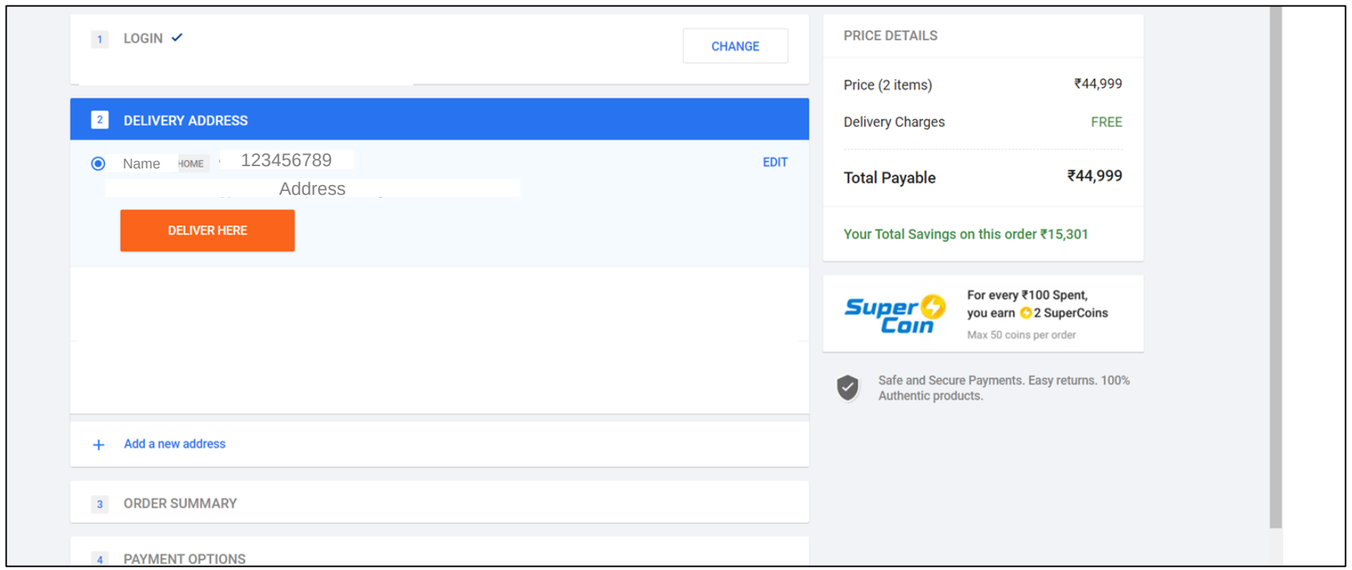
There should be an option to add a product to the wishlist for every product.

In the wishlist page, a list of products should be displayed in the form of a list, and an option should be available to place an order.

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

Checkout Story

The end user should view the checkout page to order the products with payment options.



Checkout Story

On the checkout page, there should be two sections.

- The first section should show the list of all ordered products.
- The second section should show details for payment.
- Availability of the product and delivery of the product to the selected delivery address should be checked.

Payment Story

The end user should review the payment summary before placing an order.

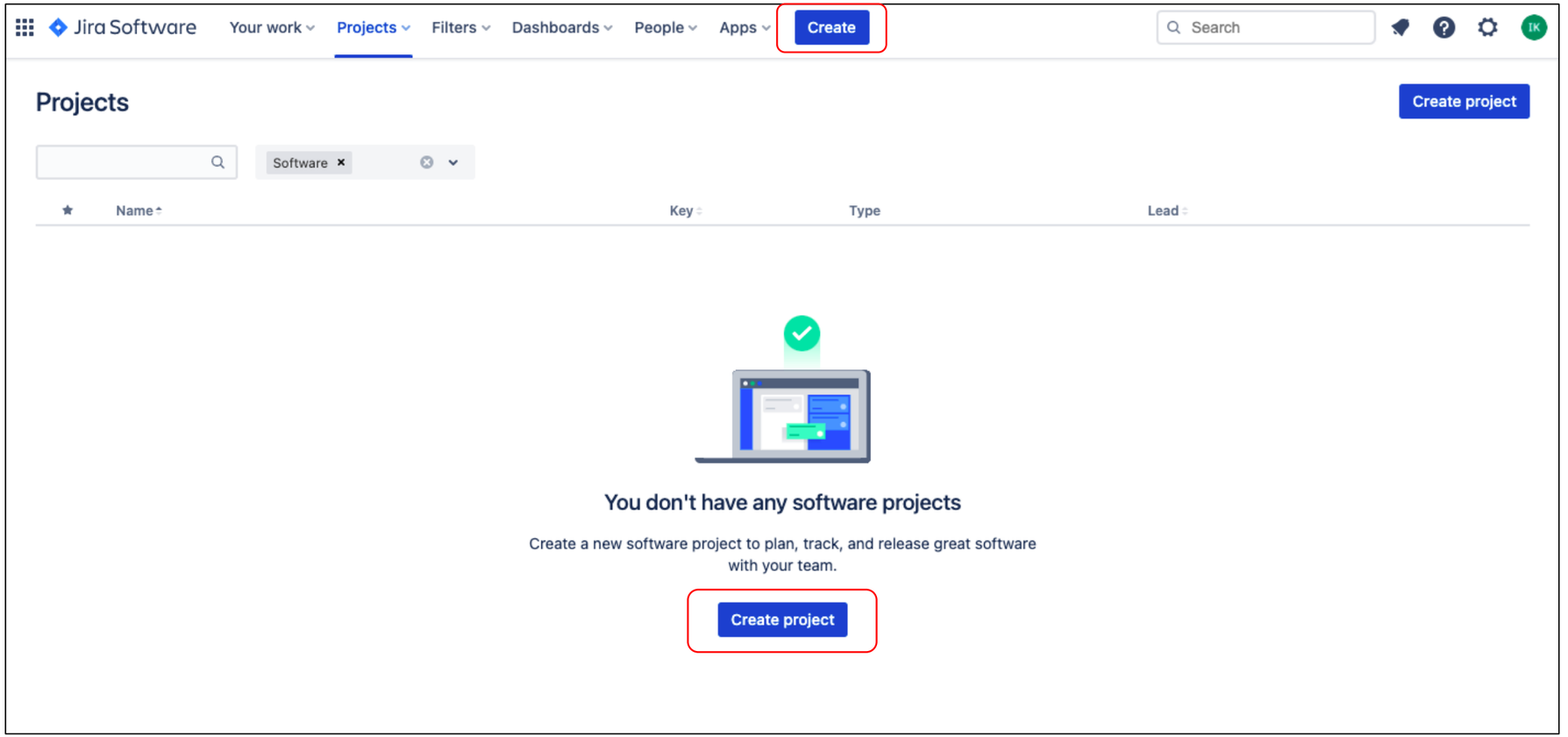
There should be a summary of orders on the payment page, with the option to change the delivery address and mode of payment.

In the online mode of payment, the payment amount should be validated.

Working with JIRA

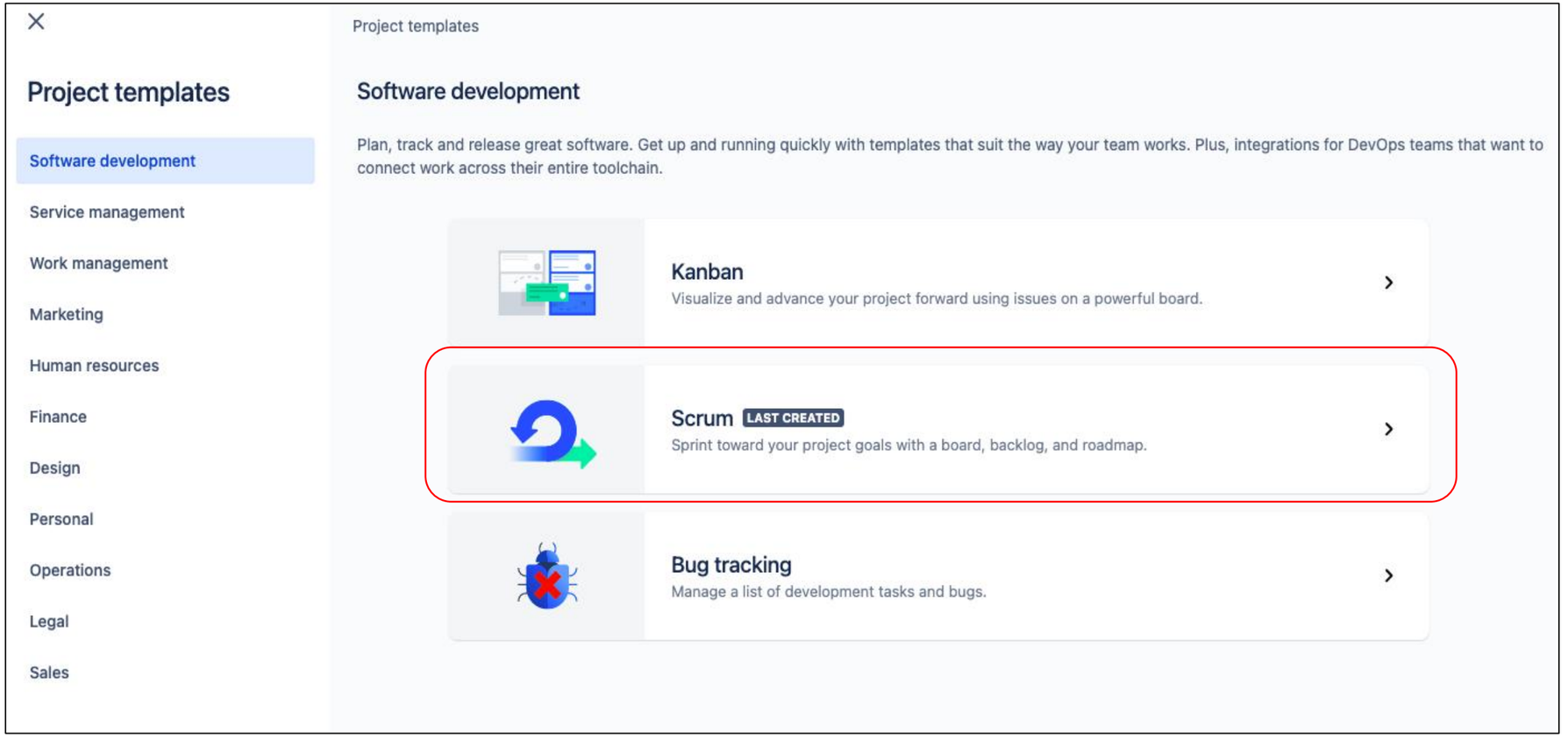
Create Project in JIRA

After creating an account in JIRA, the first step is to create a project to manage with Agile.



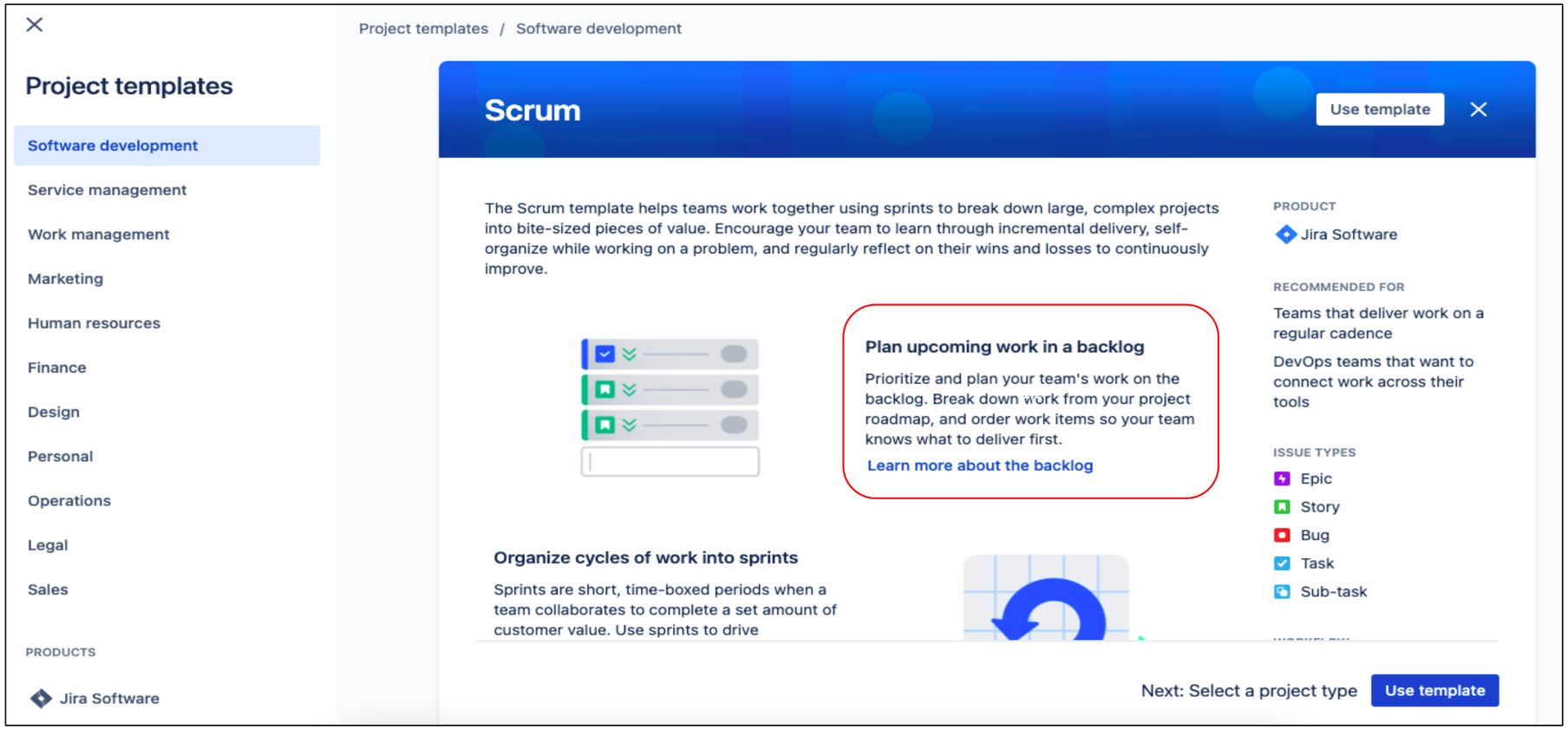
Different Project Options

When the user clicks on Create Project, three different ways to create a Project in JIRA are displayed. Scrum is preferred as it is easy to manage a project using Agile.



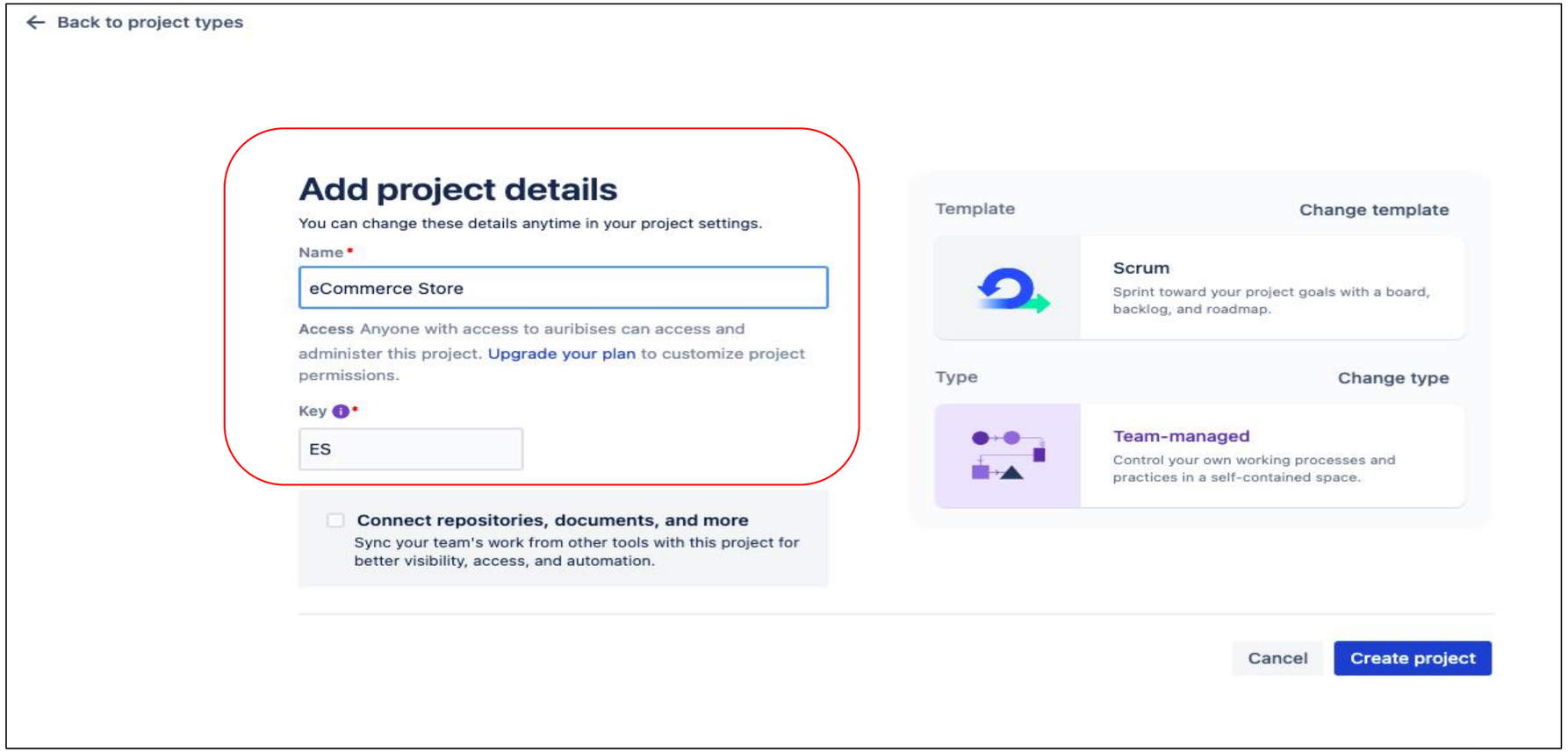
Scrum Template

With the Scrum project theme, the user can plan and create Sprints and break down a complex project into small projects.



Project Details

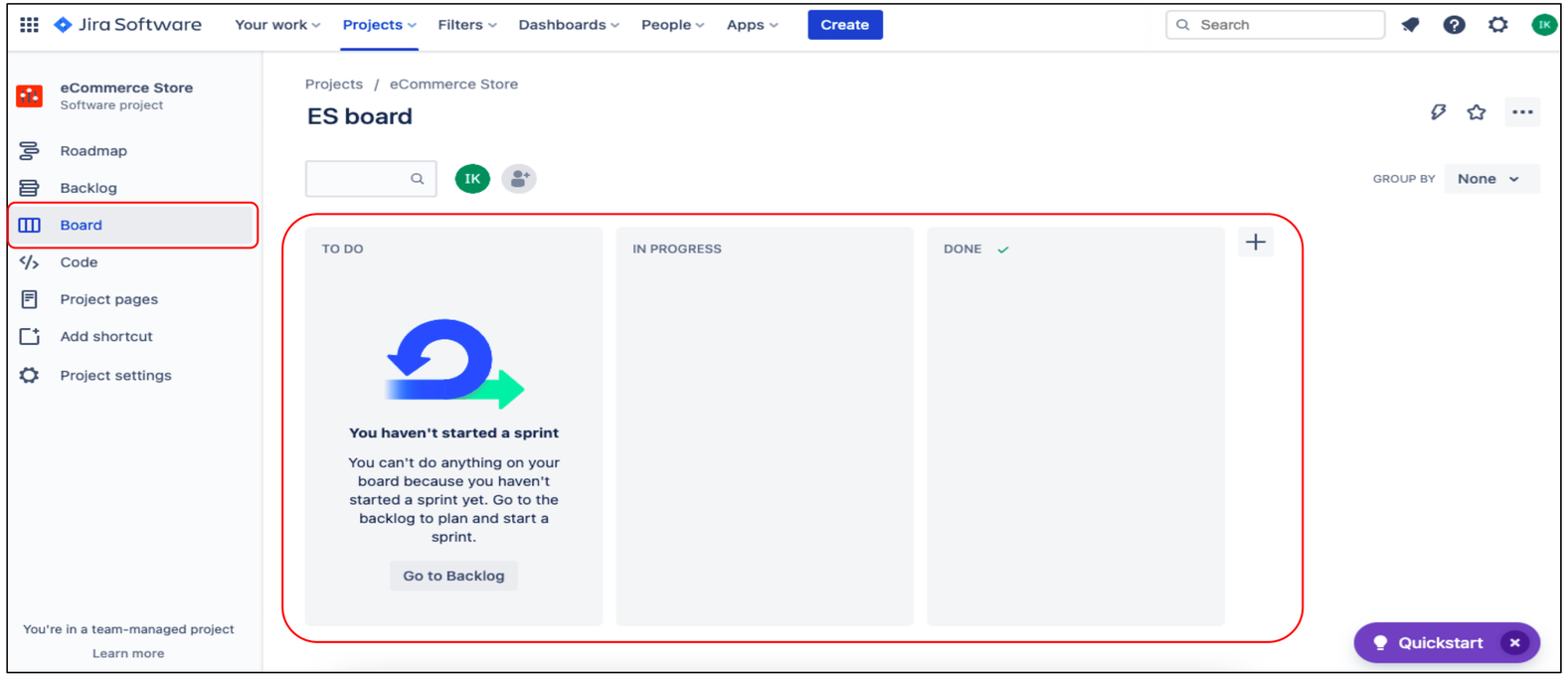
Next, the user can add details of the project, including the name of the project. A key will be auto-generated for the project, and it can be changed per the requirement.



Project Home Page

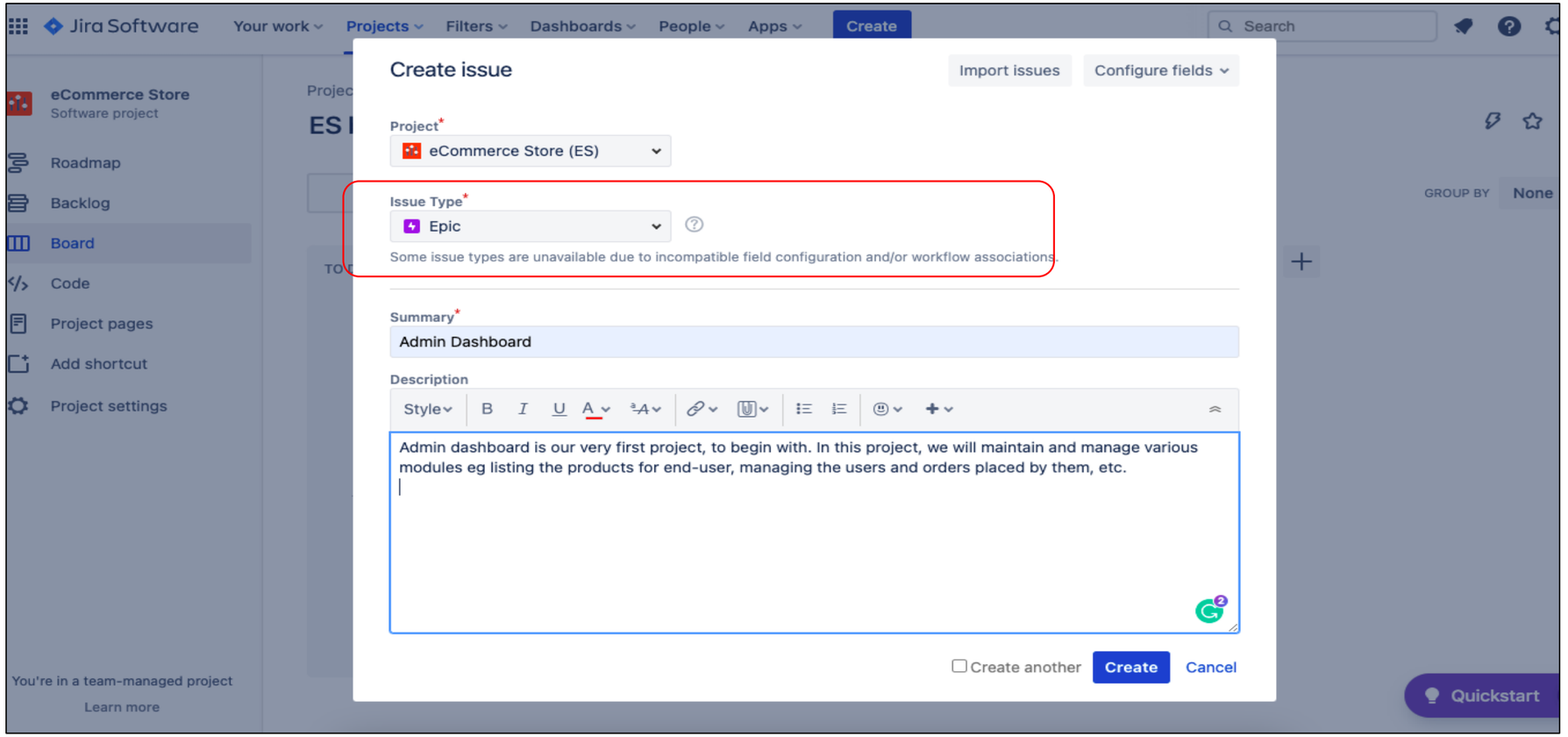
After creating and adding the project, the user is redirected to the home page with Boards Options.

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



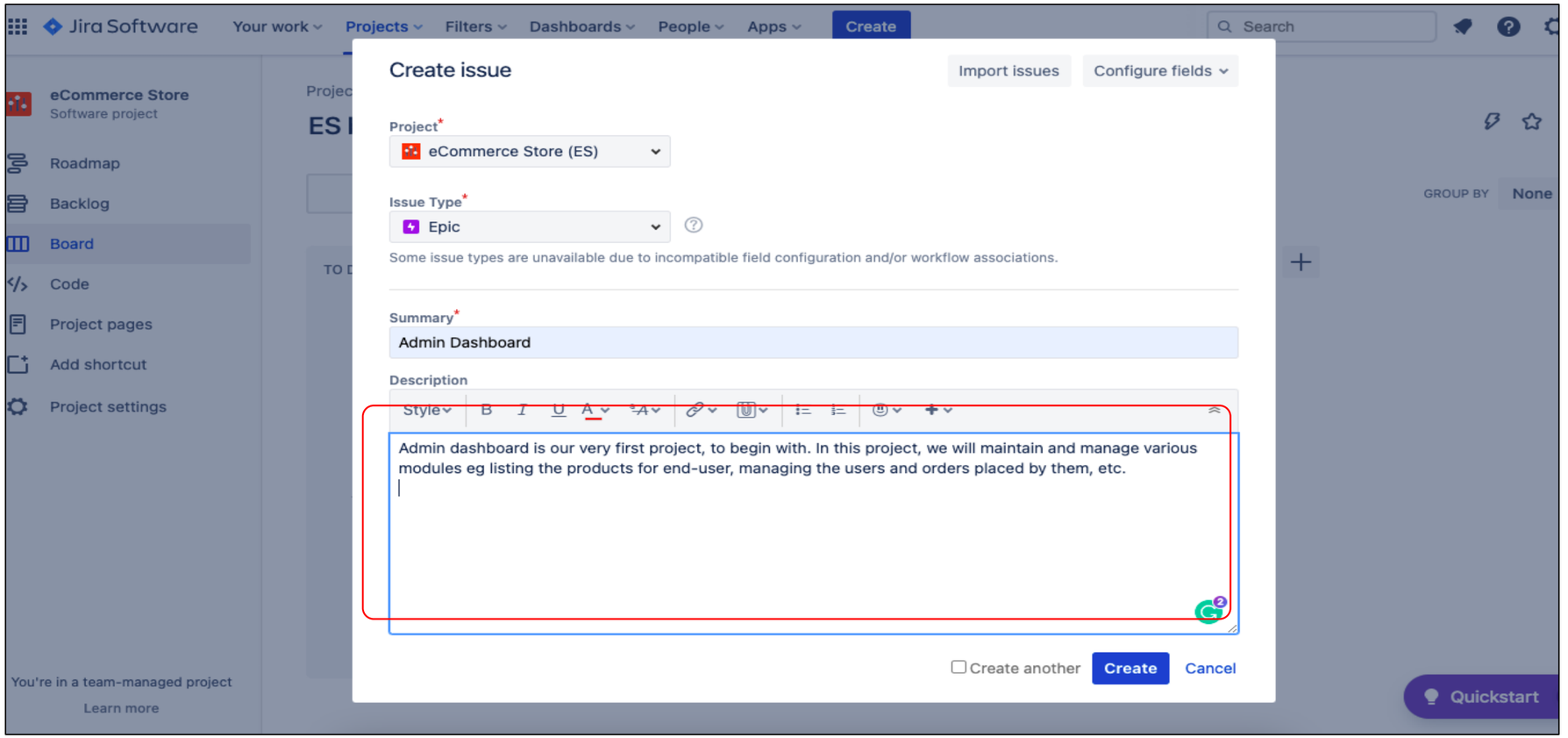
Create Epic

The next step is to create Epics. Click **Create** button on the project home page to create an Issue.



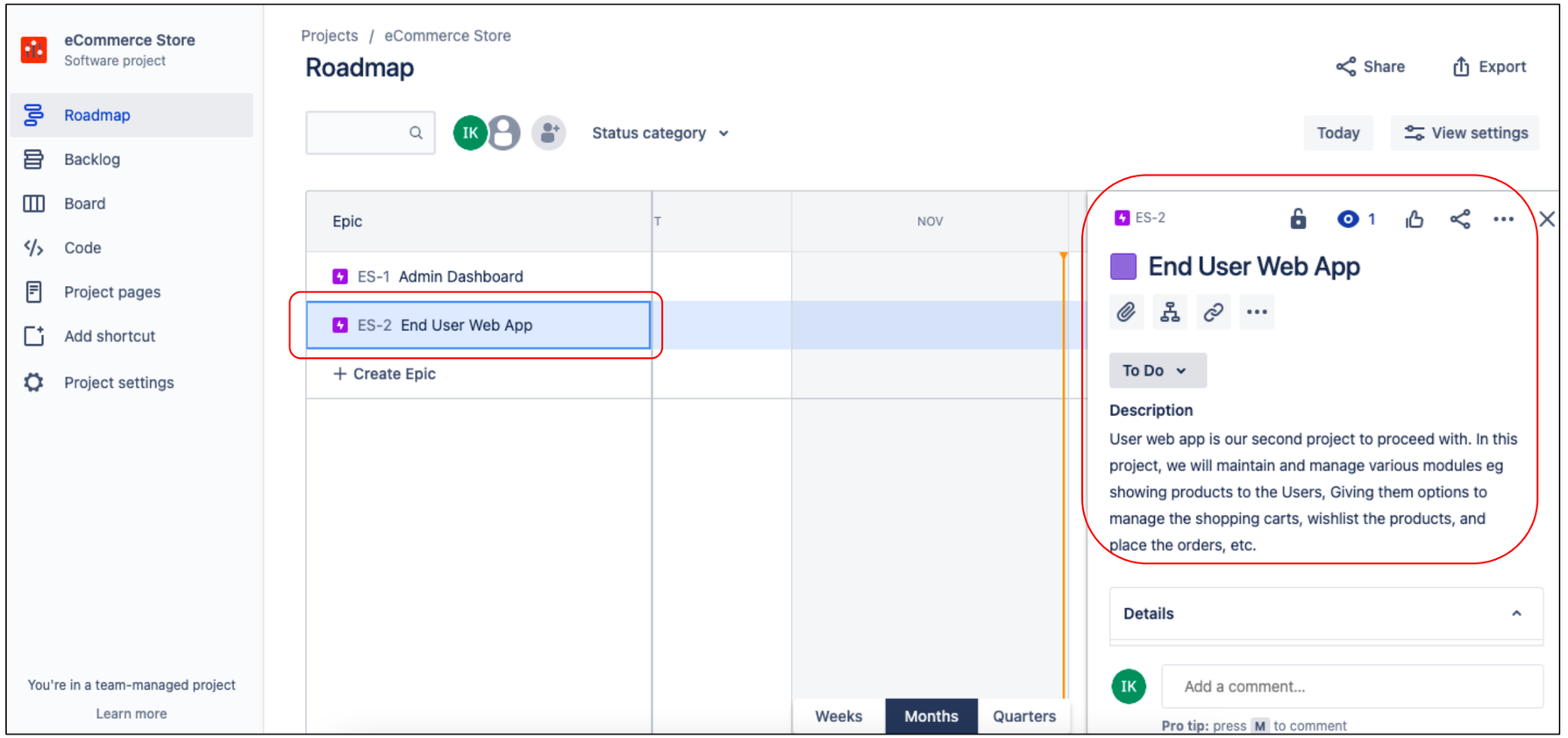
Create Epic

The issue can be Epic, Story, Task, or a Bug. Select Epic in Issue Type and write a summary and detailed description.



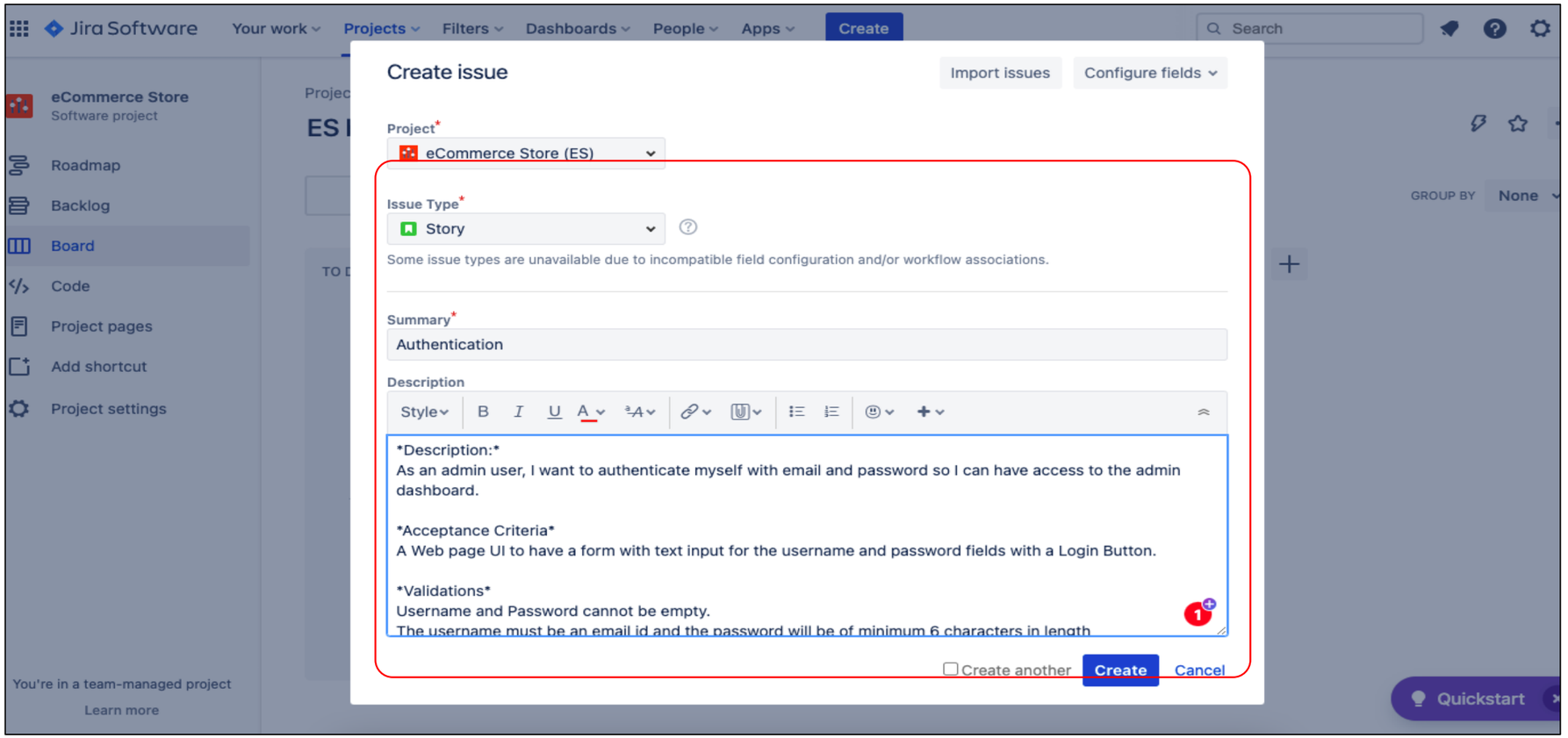
Road Map

The user can view added Epics in the Projects Roadmap Option. Moreover, when the user selects any epic, its summary and description can be seen.



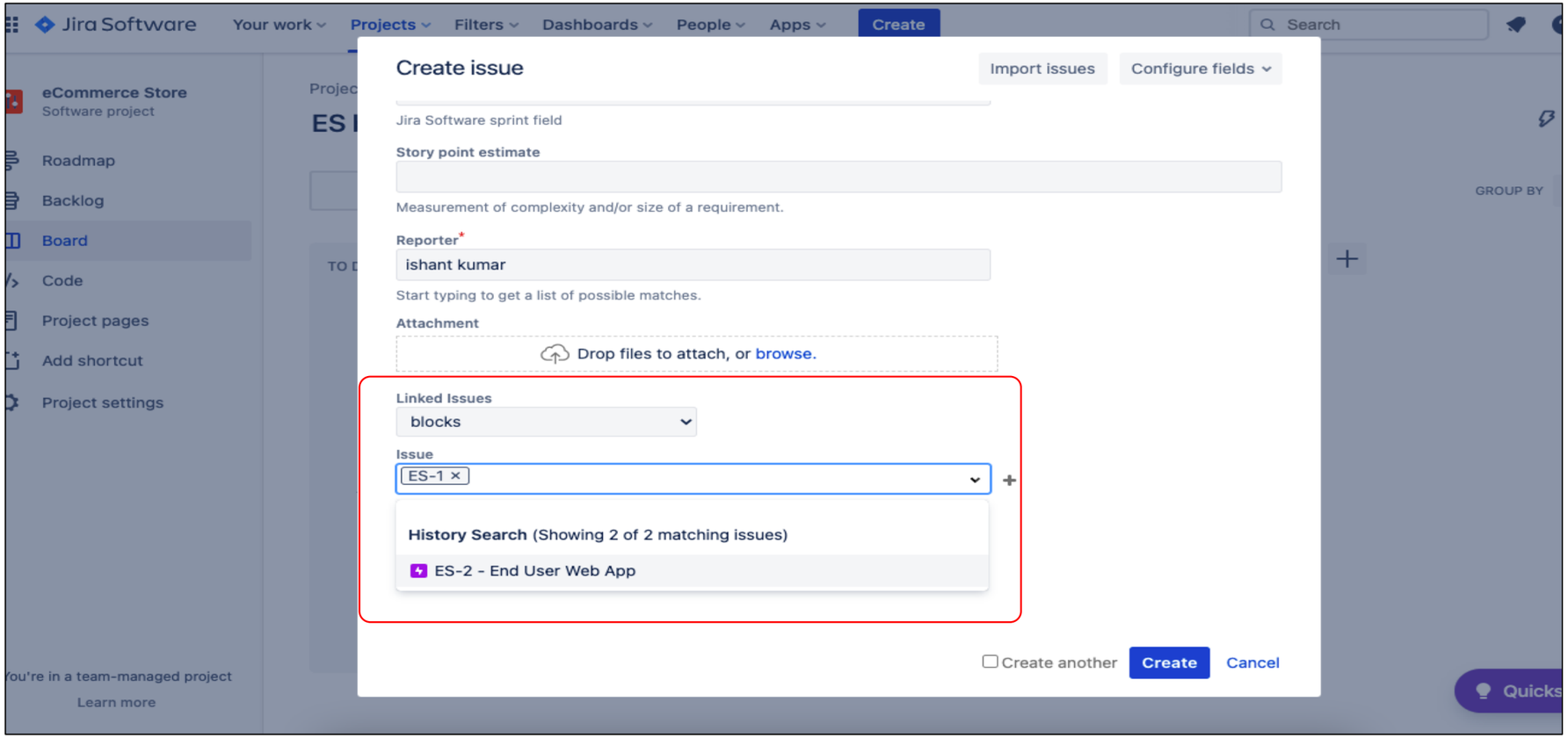
User Story

Similarly, the user can create an issue type as Story and write complete details in the description section.



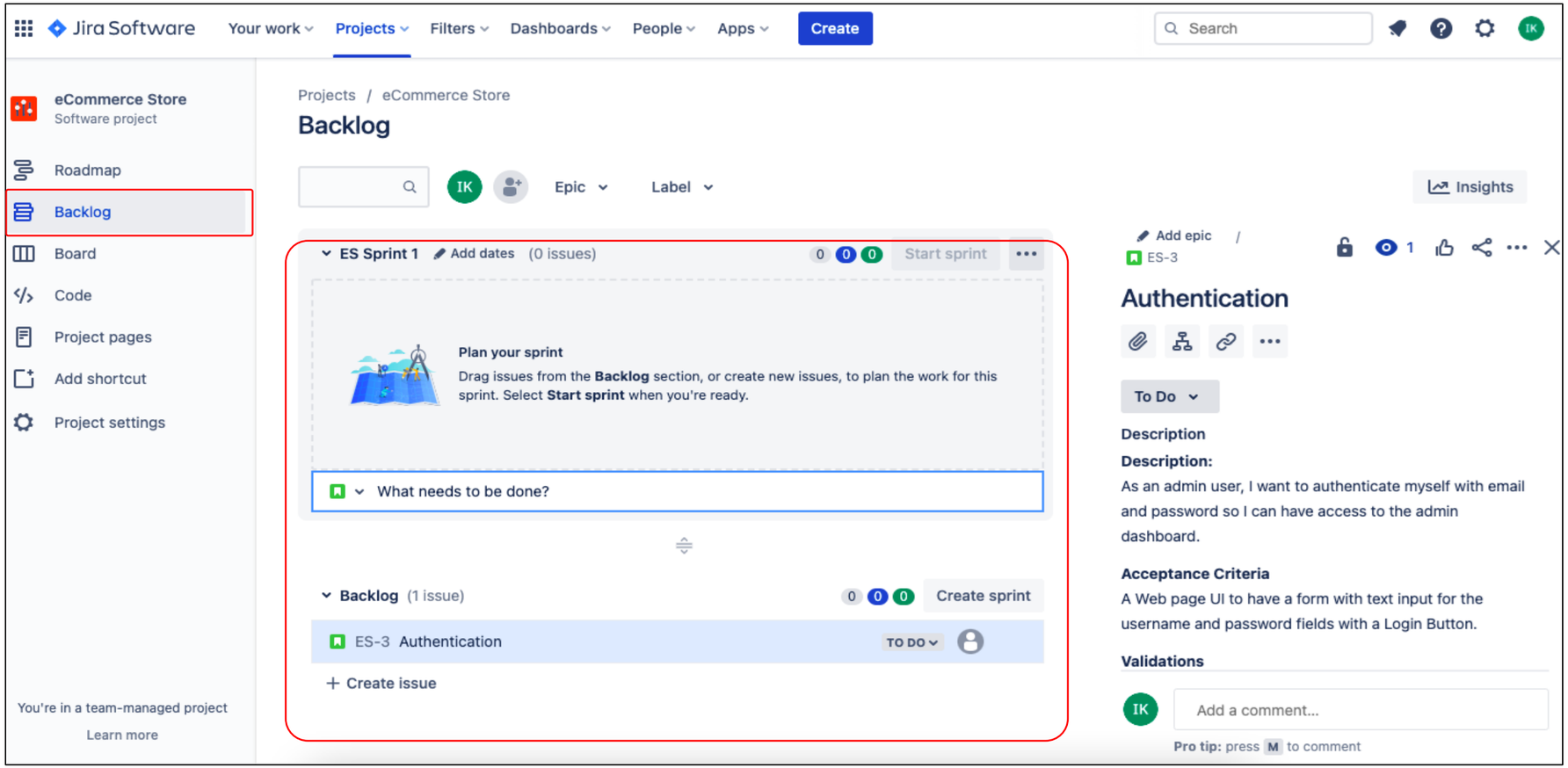
Link Story to Epic

While creating the story, an option to link it to the Epic is also available in the dropdown.



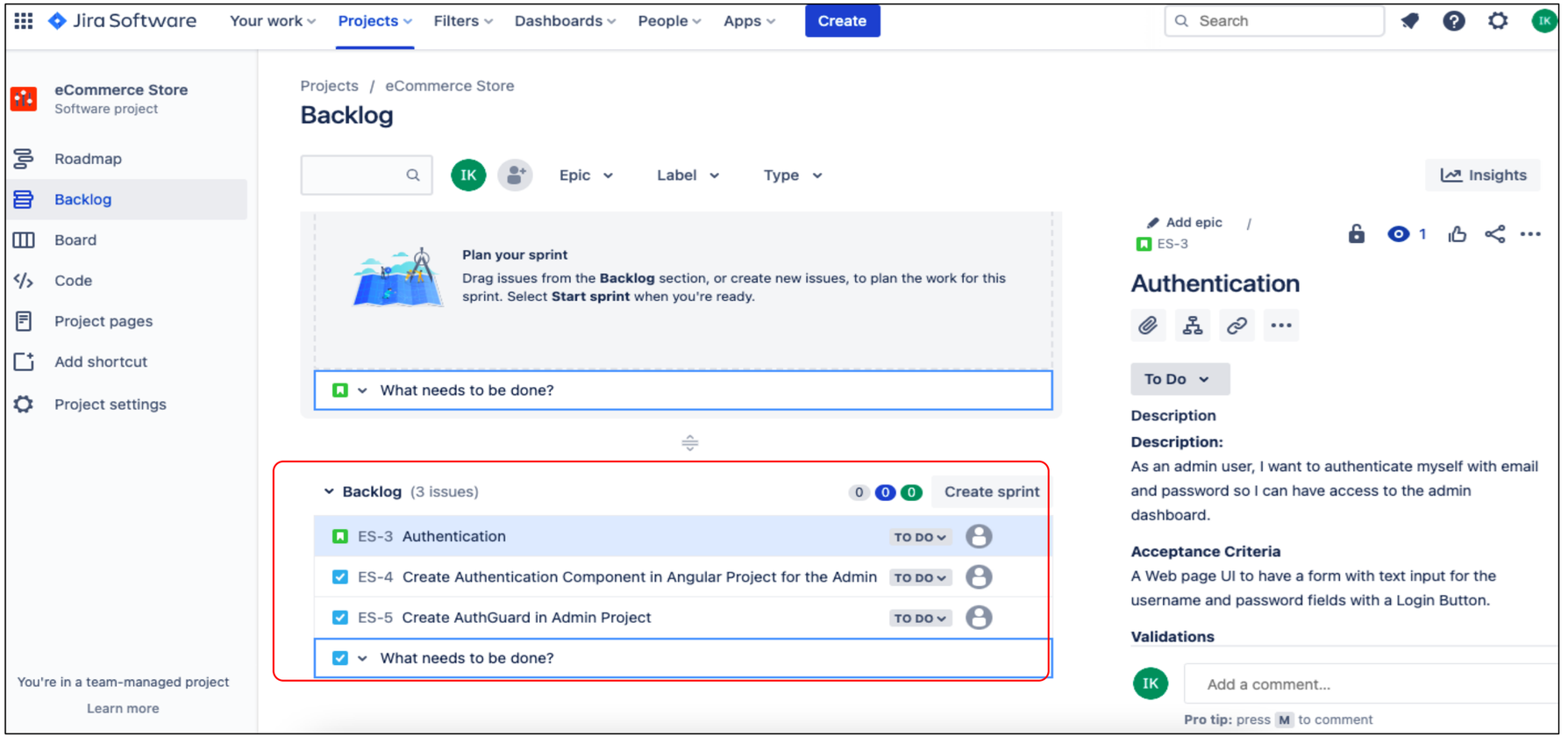
Backlog

After successfully creating the Epic story, it is listed as a Backlog in the JIRA Project Dashboard. Now, the user can start writing technical tasks to implement the User story.



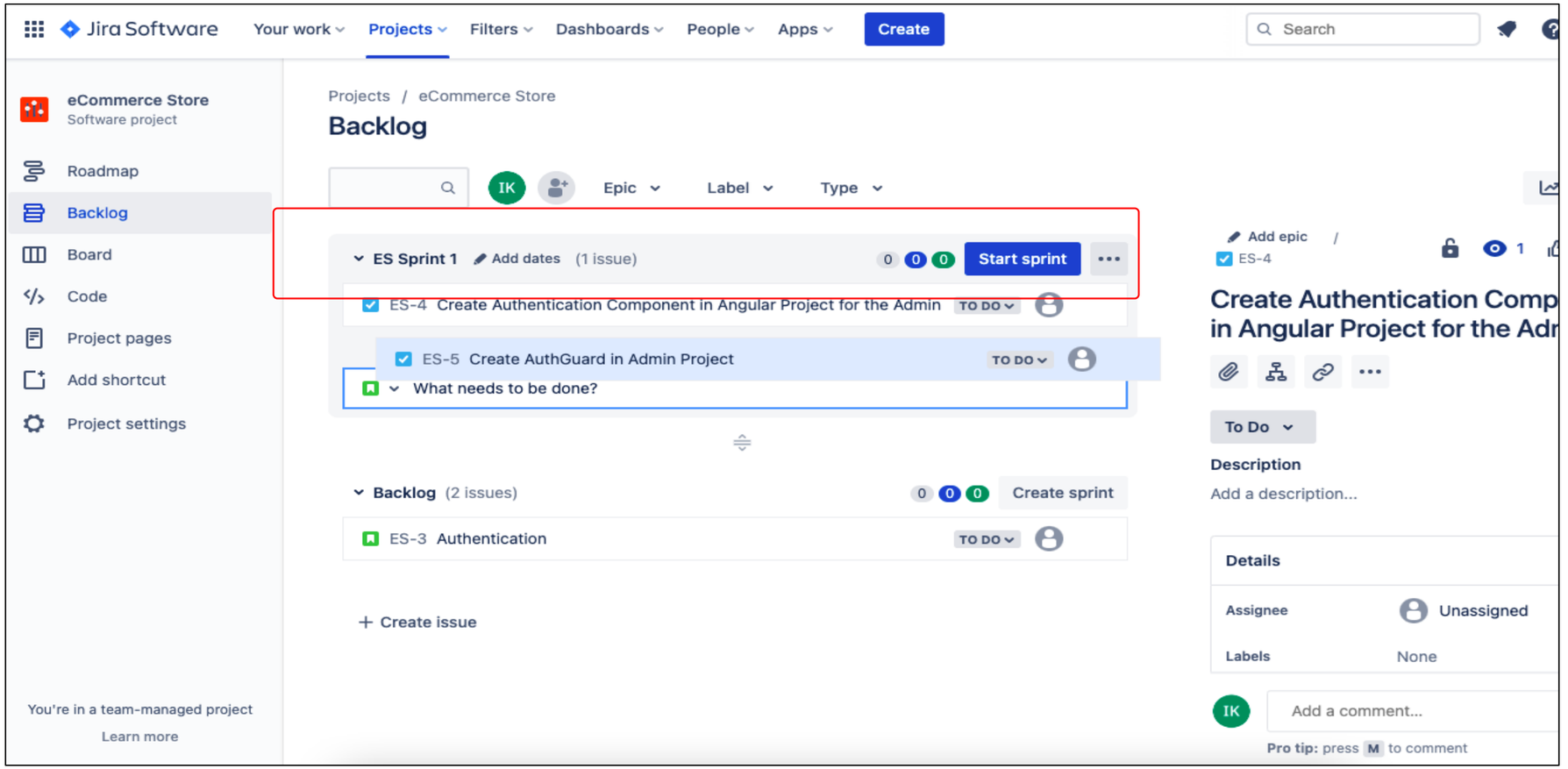
Tasks

JIRA asks to create tasks in a great UX manner. Users can continue creating tasks and assign them to the team member.



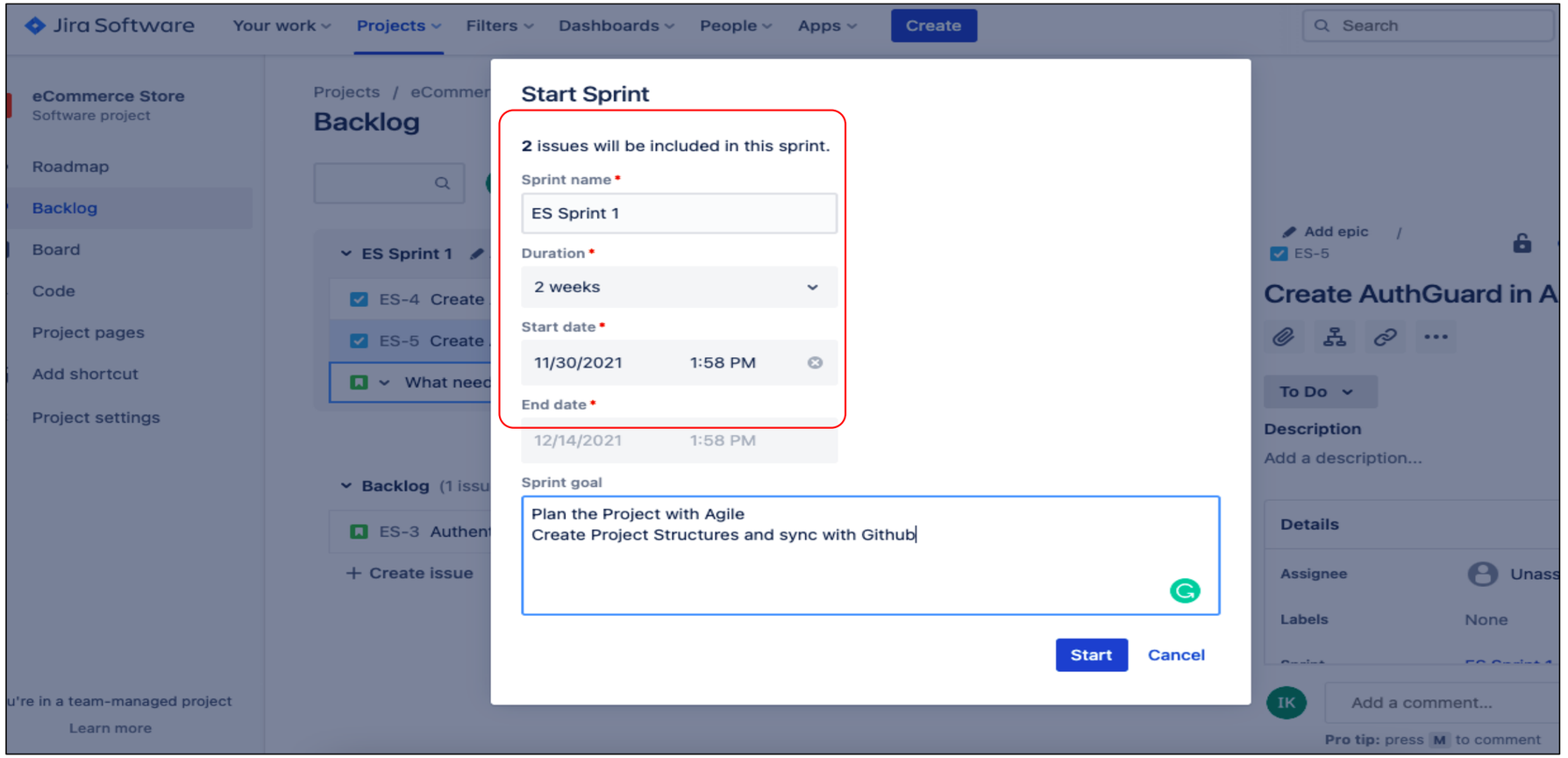
Sprint Planning

By default, a Sprint in the backlog section is displayed by the name Sprint 1, in which tasks can be added and planned.



Starting a Sprint

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



Caltech | Center for Technology & Management Education

Create Project Structure and Sync with GitHub

Powered by simplilearn

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

Creating a New Angular Project

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

Creating a New Angular Project

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 below command in command shell.

```
ng new Admin-Dashboard
```

Following 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 is 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)
```

Creating a New Angular Project

Command	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.

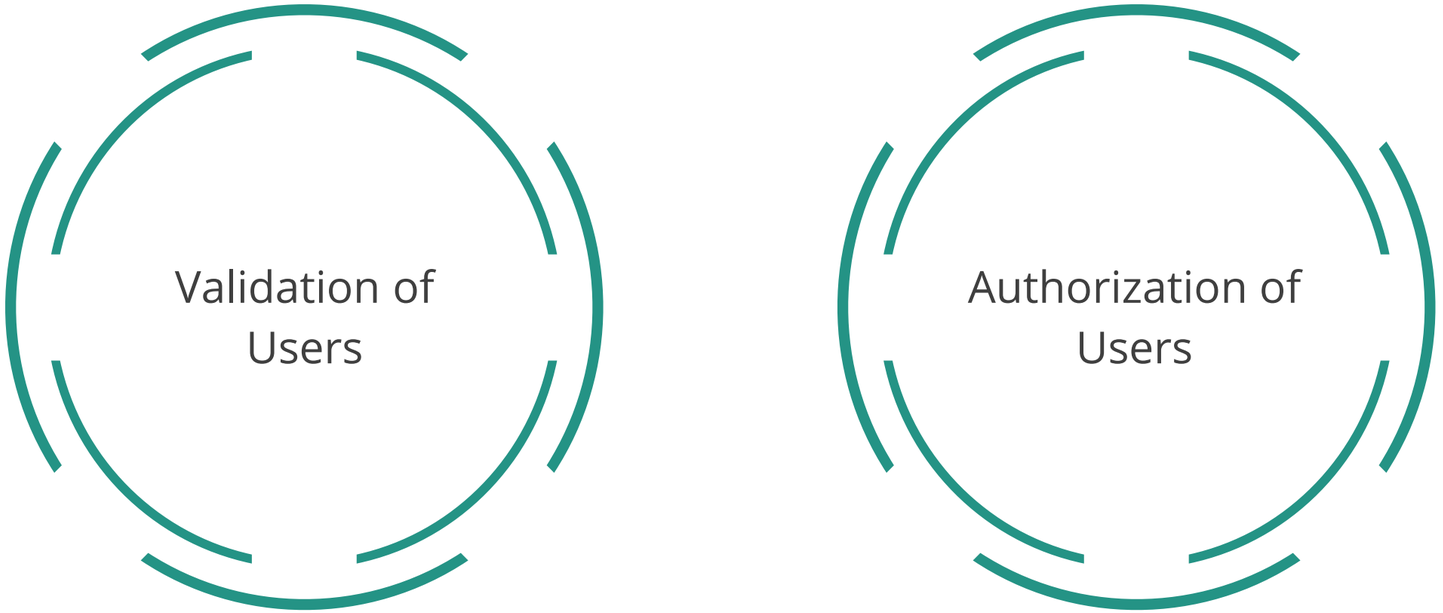
Creating a New Angular Project

Command	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:



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.

Optionally, CSS styles are applied to the template.

Generating Auth Component

For terminal change directory to user project root folder use this command:

```
cd admin-dashboard
```

Then, execute the below command:

```
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 Component

On terminal change directory to user project root folder using the command:

```
cd admin-dashboard
```

Then execute the below command:

```
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/aut
```

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 will 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 this 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

In order to view the products, generate product-list component in products folder using this 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
```

Web Admin Dashboard: Users Module

Generating User Component

Generate user component using this 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 this 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 to 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 this 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 is shown.

It implements management of order transition from warehouse to user's doorstep.

It updates order details and notifies the customer.

Generating Shipment Component

Generate shipment component using this 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
```

Generating Shipment Component

Generate shipment component using 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 init 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:
```

On branch master, changes to be committed: use "git restore --staged <file>..." to unstage:

```
modified:   src/app/app.module.ts
      new file:   src/app/auth/auth.component.css
      new file:   src/app/auth/auth.component.html
      new file:   src/app/auth/auth.component.spec.ts
      new file:   src/app/auth/auth.component.ts
```

Git init 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 this command:

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

Git Push Command

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

Use below command:

```
git push origin master
```

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

Creating a New Angular Project

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

Showing products to the users

Giving users options to manage the shopping carts, wishlist the products, place the orders, etc.

Creating a New Angular Project

Create the admin dashboard by executing 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 as 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 this 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 this 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 Gaurd

Open project in the terminal, then use this 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 this 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 contains the essential details of the user who has registered on the web app.

It includes basic information such as:

Name, email, phone number, and optionally profile image

Management of delivery addresses

Generating Profile Component

Open project in the terminal, then use this 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 includes the product listing.
- The product list is 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 this 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 Product Component

The shopping cart component shows the user the products added to the cart to purchase.

Checkout option will 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 will be available to place an order at the checkout time.

Generating Shopping Cart Component

Open project in the terminal, then use this 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 this 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 init 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:
```

On branch master, changes to be committed: use "git restore --staged <file>..." to unstage

```
modified:   src/app/app.module.ts
      new file:   src/app/auth/auth.component.css
      new file:   src/app/auth/auth.component.html
      new file:   src/app/auth/auth.component.spec.ts
      new file:   src/app/auth/auth.component.ts
```

Git init 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 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 below command:

```
git push origin master
```

Key Takeaways

An admin user 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

- An admin user 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 block 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.

Before the Next Class

Since you have completed this session successfully, for our next discussion you should:

- Review important HTML tags for creating web pages
- Review the forms in HTML
- Check with CSS for styling the HTML forms
- Get hands-on with various SQL commands for CRUD operations
- Review how to create tables in MySQL

What's Next?

Now, we have finished designing and developing web pages for the admin dashboard in our Angular project. In our next discussion, we will:

- Work on Authentication Page, Users Page, Orders Page, and others to create the UI for the project
- Work on deciding the various attributes for the tables in MySQL
- Create the structure of tables in MySQL which will store data from the web pages