EKO INNOVATION & TRADING S.L.
E-COMMERCE WEBSITE PROJECT

TEAM MEMBERS

Number	Name	Role
220104004829	Zerda Gül	Scrum Master/ Back-End Developer
220104004088	Dicle Çoban	Product Owner/Front-End Developer
210104004033	Mehmet Emir Albayrak	Back-End Developer
220104004038	Semih Sarıkoca	Back-End Developer
220104004928	Anhelina Bondarenko	Front-End Developer

User Stories and Related Scenarios of the Project

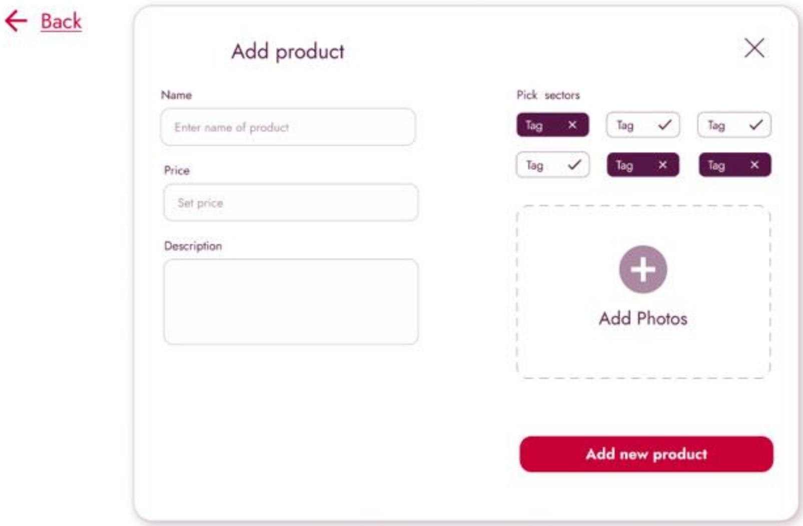
User Story #1: Product Management

Ahmet works at EKO Innovation & Trading S.L., a company active in the smart city, energy, ITS & Traffic, security, and iron industries. He is responsible for selecting and managing the products to be sold on the EKO Innovation & Trading S.L. website. Ahmet envisions a website that showcases the company, includes explanations of relevant industries, and lists products for each sector. He wants the ability to add products, remove them, and update information about them, ensuring that the website adapts whenever the company's product lineup changes.

Related Scenarios for User Story 1# Product Management

Scenario #1: Product Addition

Initial Assumption: Ahmet, who serves as the Product Admin for EKO Innovation & Trading S.L., successfully logs into the company's website platform and accesses the product management dashboard. He has the necessary permissions to add, edit, and manage product listings. Ahmet is ready to add a new product that is set for sale on the website.



Normal: Ahmet goes to the "Products" section within the dashboard, where he can manage current products and add new ones. He clicks the "Add New Product" button to begin the product creation process. He fills out all mandatory fields. He specifies the name of the new product, adds a detailed description outlining the product's features and usage, the selling price, inputs the stock quantity available in inventory, chooses the appropriate sector (e.g., smart city, energy, ITS & traffic) to categorize the product. Ahmet uploads an image of the product, ensuring it meets file requirements (e.g., resolution and file size limits). After filling the name, description, price, stock, sector, image information, he clicks "Add new product" to finalize the entry. A pop-up or on-screen message appears confirming that the product has been successfully added. The newly added product is immediately visible to website customers in the "Products" section, complete with all entered details.

What Can Go Wrong:

- If Ahmet omits any mandatory fields (like product name or price), the system immediately highlights the missing information and prompts him to complete it before saving.
- If the uploaded image file exceeds the allowable size or format, an error message appears, and the system prevents saving until a suitable image is uploaded.
- If Ahmet attempts to add a product with an identical name, the system may prompt him to review and confirm the entry to prevent duplicates.

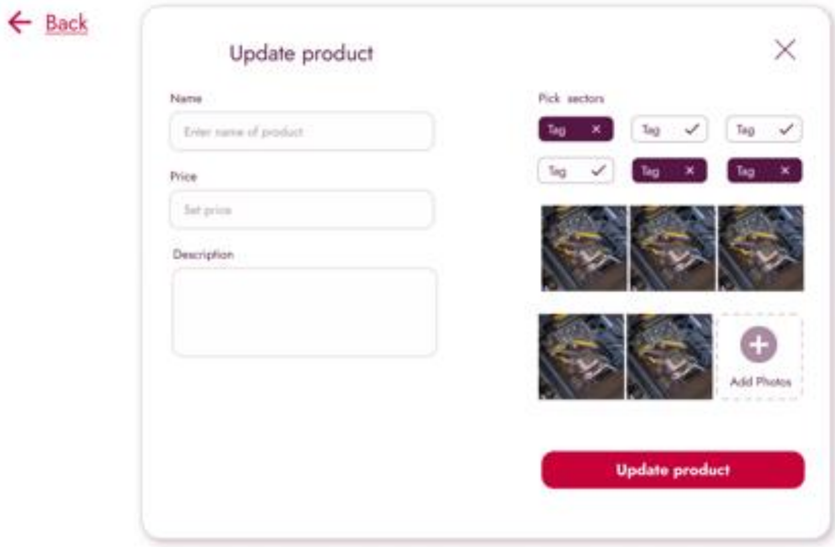
Other Activities: Another product admin uses the website functionally at the same time.

System State on Completion: Upon successful completion, the new product is added to the database and immediately displayed to users on the website. The product's data, including its photo, name, description, price, stock status, and sector classification, is accessible in website.

Scenario #2: Updating Product Information

Initial Assumption: Ahmet, the Product Admin at EKO Innovation & Trading S.L., identifies that a product's details (such as name, description, price, or stock status) are outdated or inaccurate. He is logged into the website's product management dashboard with permissions to modify product information.

Normal: Ahmet navigates to the "Products" section, where he can view all the items currently listed for sale on the website. He finds the specific product that requires information updates. Ahmet clicks the pencil icon next to the product he wants to modify. This opens a form where he can update the product's details. Ahmet makes necessary changes to fields like name, description, price, stock, sector or new image. He clicks "Update product" to save the modifications. The system displays a success message confirming that the product information has been updated. The changes are immediately reflected on the database. Website is updated and customers see the latest information.



What Can Go Wrong:

- If a technical issue occurs (e.g., server downtime or connectivity problems), the system may not save Ahmet’s changes.
- If Ahmet enters incorrect or invalid values, such as a negative stock number or invalid characters in the price field.
- If customers are actively viewing the product during the update, they might see inconsistent information temporarily.

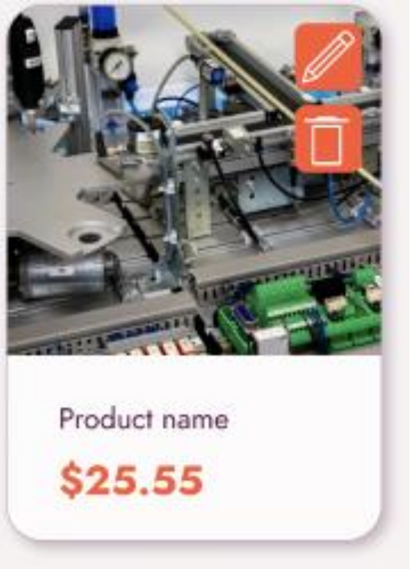
Other Activities: Another product admin uses the website functionally at the same time.

System State on Completion: Upon successful completion, the product information is fully updated and synchronized with the website’s frontend. Customers now have access to the accurate, up-to-date product details, ensuring transparency in product offerings. Ahmet can review the product in the “Products” section to confirm the successful update. The system maintains an accurate product catalog, supporting customer trust and inventory management.

Scenario #3: Removing a Product from Sale

Initial Assumption: Ahmet, the Product Admin, has identified a product on the EKO Innovation & Trading S.L. website that needs to be removed from sale, either due to stock unavailability, discontinuation, or a strategic decision. He is logged into the website’s product management dashboard and has permission to delete products.

Normal: Ahmet navigates to the “Products” section, where he can view the full list of items currently listed for sale on the website. Ahmet clicks the trash bin icon located in the corner of the specific product card he wishes to remove. A pop-up confirmation dialog appears, warning that deleting this product will permanently remove it from the website and make it unavailable to customers. He clicks “Confirm” in the dialog to complete the deletion process. A confirmation message appears, notifying Ahmet that the product has been successfully removed from the listing. After confirming deletion, the product immediately disappears from the “Products” section and is no longer visible to customers browsing the website.



What Can Go Wrong:

- If a temporary system error or glitch occurs, the product may not be deleted, resulting in it remaining visible on the website. In such cases, an error message should appear, informing Ahmet that the deletion was unsuccessful and prompting him to try again later or contact technical support.
- If there’s a delay in syncing the website database, the product may temporarily appear on the front end even after deletion.

- Customers currently viewing or interested in the removed product might see an error or “Product Unavailable” message if they attempt to add it to their cart or view its details. The system should ideally prevent access to deleted products in real-time to avoid confusion.

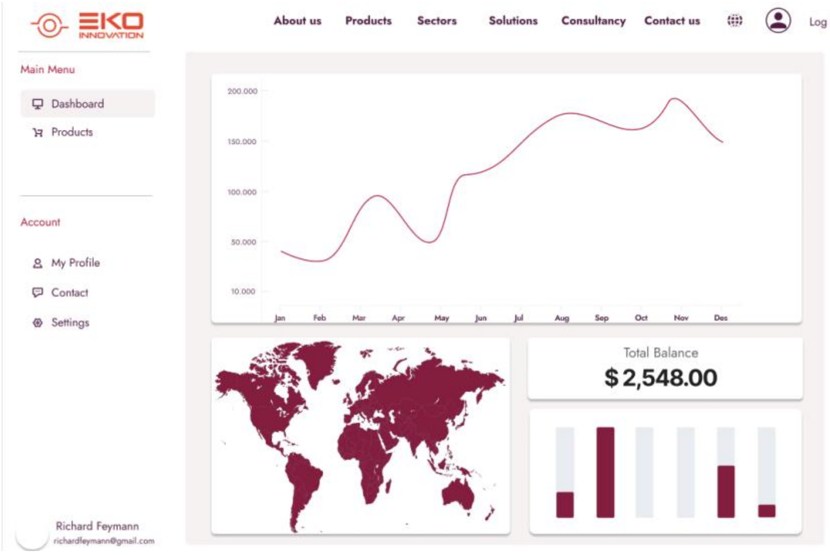
Other Activities: Another product admin uses the website functionally at the same time.

System State on Completion: Upon successful completion, the selected product is removed from the website’s database and is no longer visible to customers. Ahmet can review the updated “Products” section to confirm the removal. The system reflects an accurate, real-time inventory for customers, maintaining the reliability of available listings.

Scenario #4: Viewing Sales Statistics

Initial Assumption: Ahmet, as the Product Admin at EKO Innovation & Trading S.L., wants to monitor and analyze sales performance for better inventory and business decisions. He is logged into the website’s product management dashboard with permissions to view detailed sales data and statistics.

Normal: Ahmet logs in and navigates to the “Dashboard” section, dedicated to displaying analytical insights and performance metrics. The dashboard displays a graph or chart that shows monthly sales figures, enabling Ahmet seeing the total balance. Ahmet can see countries of sales, allowing him to identify key markets and target regions. After reviewing the statistics, Ahmet notes any patterns or areas for improvement, such as adjusting inventory based on high-demand regions or exploring marketing opportunities in underperforming areas.



What Can Go Wrong:

- If the data has not been updated or synchronized with the latest sales records, Ahmet may view inaccurate or stale information.

Other Activities: Another product admin uses the website functionally at the same time.

System State on Completion: Ahmet successfully reviews and interprets up-to-date sales statistics on the dashboard. These insights provide Ahmet with valuable information to make informed decisions on inventory, marketing, and sales strategies, enhancing overall business performance.

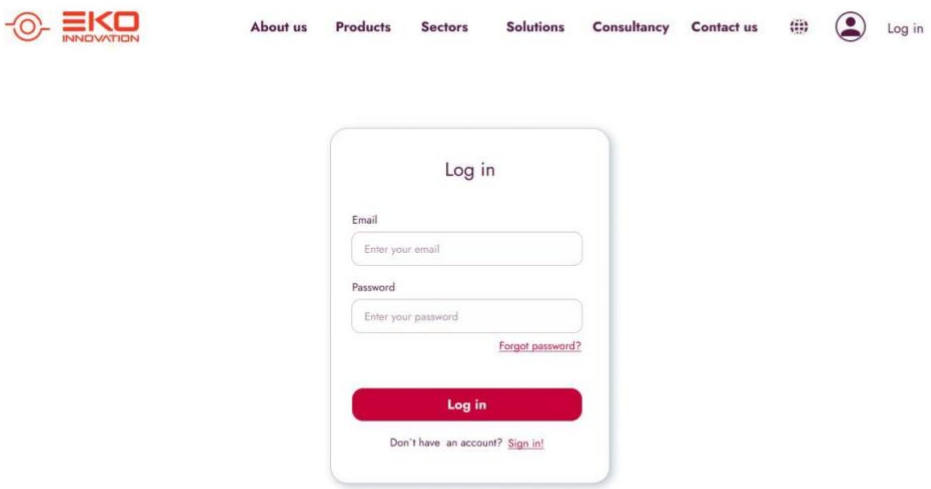
User Story #2: Order Request

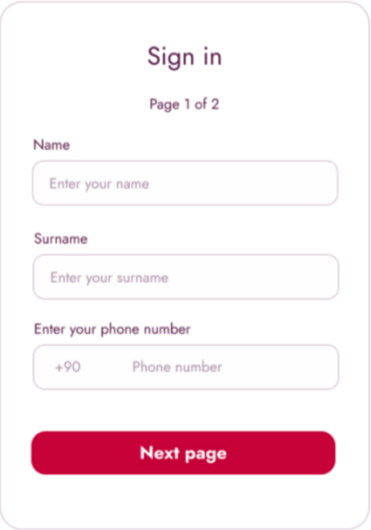
Zehra is an employee responsible for sourcing products needed by her company. She relies on the EKO Innovation website for streamlined purchasing transactions between companies. She seeks a fast, reliable and secure shopping experience to procure the products her company requires as efficiently as possible. Zehra must create an account and log in to place an order on the EKO Innovation platform. Then she can add products to her cart.

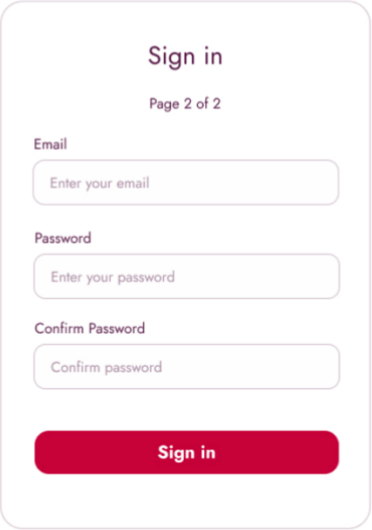
Related Scenario #1: Place an Order

Initial Assumption: Zehra enters the domain name of the EKO Innovation website. She wants to create an account, and log into her account the starts shopping. The site provides a user-friendly interface that allows for seamless navigation and efficient product search.

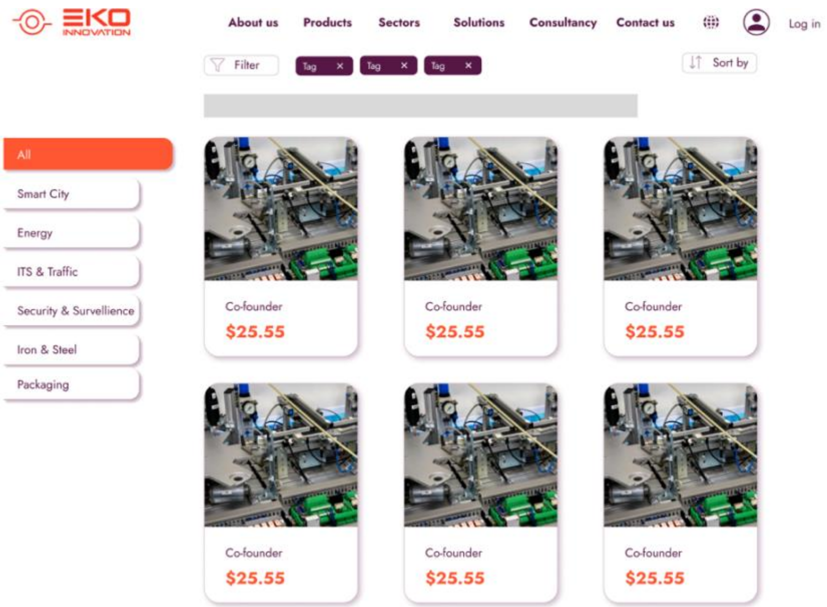
Normal: Zehra needs to create an account on the EKO Innovation website. She navigates to the login page by clicking "Log in" button. Zehra needs to create an account that's why she clicks "Sign in!" text under the button in login page. She enters necessary details like her name and surname, email address, phone number, password and confirm password. After submitting the form, she receives a verification email. Zehra clicks on the link in this email to verify her account, completing the registration process. Next, she logs into her account using her email and password, gaining access to her profile and the shopping functions of the platform.

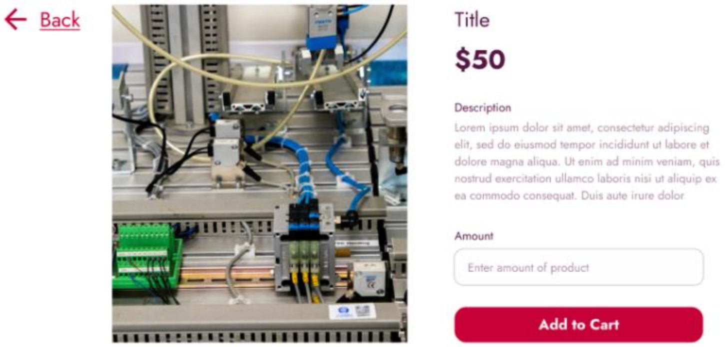






Zehra explores the website in “Products” section, brows product categories to find the required items. She clicks the specific product image, sees the information of the product. She enters the amounts based on her company’s need and adds the product to her c art by clicking the “Add to Cart” button.

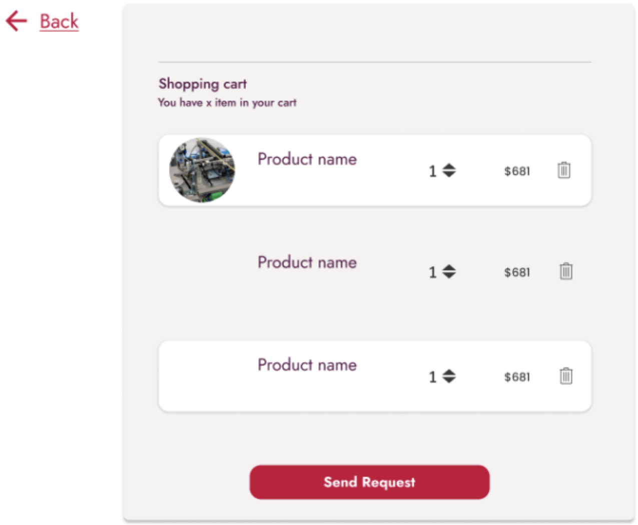




Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor

Upon clicking the “Shopping cart” button, Zehra should see a summary with individual product names, quantity and a total cost in her cart. She can see product’s name, quantity, price, and total cost Zehra reviews the items in her cart, verifying each selection, adjusting quantities, or removing unwanted items.

Zehra clicks “Send Request” to confirm her order. After successfully submitting her request, Zehra receives a confirmation message that her order has been received and is in processing. Her profile page allows her to track the order status, with real-time updates from “request” to “done.”



What Can Go Wrong:

- Server issues could prevent Zehra from adding items to her cart or completing her order.
- If Zehra’s session expires, she may need to re-log in to proceed with the order.
- If order status updates are delayed, Zehra may be unable to follow her order’s progress accurately.

Other Activities: Another product admin uses the website functionally at the same time.

System State on Completion: Upon successful completion of the order, Zehra receives a confirmation that her purchase is being processed. The system updates her profile with the order status, moving from “request” to “done” as her order proceeds through each phase. This allows her to monitor and manage the process, ensuring she receives the necessary products securely and promptly.

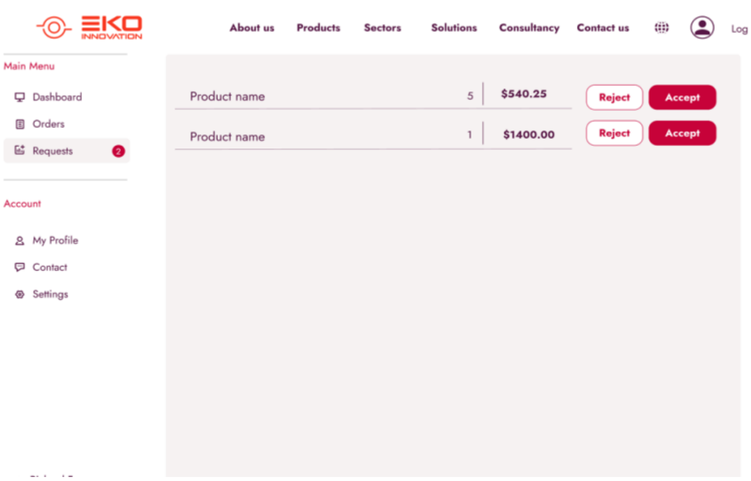
User Story #3: Order Management

As an Order Admin, Merve is responsible for viewing and reviewing all orders on the EKO Innovation platform. Merve wants to quickly approve incoming orders to meet customer requests and enhance customer satisfaction by ensuring a smooth and efficient order process.

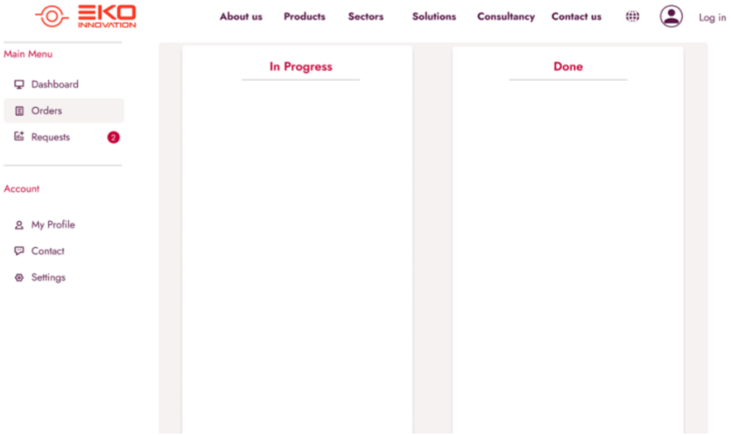
Related Scenario #1: Approve an Order

Initial Assumption: Merve is Order Admin in EKO Innovation. She has permissions to reject or approve the requests of customers and to modify order status. Merve logs into the EKO Innovation admin panel and navigates to the “Requests” section, ready to review and approve or reject orders.

Normal: Upon login, Merve sees a dashboard with all "Requests" listed, each showing an overview of order details such as customer name and total amount. Merve selects an order to review and views comprehensive information, including the list of products, quantities, total cost. If the order is complete and accurate, Merve clicks "Accept". If the order has errors or discrepancies (e.g., incorrect quantities), Merve can choose to reject it by clicking "Reject" button. The order status is updated to "in-progress". She can see the status of orders in the "Orders" section. Merve can change the order status at various stages (e.g., in-progress, done). "Done" stage is for delivered orders. Customer sees the status changings. Status updates are reflected in the customer's profile, allowing real-time order tracking. Merve should be able to view historical orders and their status.



Product name	5	$540.25	Reject	Accept
Product name	1	$1400.00	Reject	Accept



What Can Go Wrong:

- If the admin panel has slow loading times, it may hinder Merve's ability to review orders efficiently.
- Merve might mistakenly reject or approve an order, especially if she's managing a high volume of orders at once.
- If Zehra's session expires during the shopping process, she may be required to re-login to continue with her order.

Other Activities: Another admin might be reviewing and approving orders simultaneously, requiring the system to handle potential concurrent updates smoothly.

System State on Completion: Upon successful completion, Merve has reviewed and approved (or rejected) all new orders, and customers can see their order status. The order information the system is up to date, with completed transactions accurately reflected in both the admin and customer interfaces.

PRODUCT BACKLOG

Epic	Key	Description	Reporter	Priority	Status
Orientation	SOF-2	This part is about getting familiar with the framework and programming languages we’ll work with. It consists of: Frontend: react.js Backend: .NET API: Rest API Github Jira MYSQL	Dicle Çoban	Highest	Done
User and Database Management / Security	SOF-11	This epic ensures secure user account controls and database protection. It covers user registration, authentication, role-based access, and encrypted storage, along with secure database management practices like schema design, access control, and backups. Together, these features enhance data security and user trust.	Dicle Çoban	High	In Progress
Shopping Cart and Product Page	SOF-17	This epic covers functionalities for adding, viewing, and adjusting items in the cart, as well as a secure checkout and payment process. It ensures a smooth and reliable shopping experience from cart to order confirmation.	Dicle Çoban	Highest	To Do
Contact and Frequently	SOF-25	This epic focuses on providing customers with easy access to support and	Dicle Çoban	Low	To Do

Asked Questions		essential information. It includes a contact form for reaching out to the support team and a well-organized FAQ section where users can find answers to common questions. Together, these features enhance user experience by offering quick assistance and self-service options.			
Control Panel and Management	SOF-28	This epic provides administrators with a centralized control panel to manage key aspects of the platform. It includes functionalities for product management, order tracking, customer account oversight, and inventory control. These tools enable efficient day-to-day operations, ensuring that administrators can maintain and update the platform effectively.	Dicle Çoban	Highest	To Do
Home Page and About Us	SOF-33	This epic aims to create a welcoming first impression with a Home Page that highlights featured products and key navigation, and an About Us section that shares the company’s mission and values, building trust and encouraging further exploration.	Dicle Çoban	High	To Do
Test Cases and Error Handling	SOF-196	This epic ensures platform reliability by implementing test cases to verify functionality and robust error handling for managing issues gracefully, providing a stable and user-friendly experience.	Dicle Çoban	Highest	To Do

Orientation

Tasks	**Key**	**Description**	**Status**
Dicle Çoban: Learning React and Figma Design	SOF-6	**Task**: Set up the React project and create basic components (navigation, header, footer, placeholder pages). **Goal**: Establish the skeleton structure of the application, set up page routing, and build the foundation for language selection. **Output**: A functional React project with placeholder pages and working language options. **Task**: Design the general layout, user interface, and interactions for all pages. **Goal**: Create mockups for both the user and admin interfaces and design a prototype for all screens. **Output**: A completed design document in Figma, ready for further development.	Done
Semih Sarikoca: Learning ASP.NET	SOF-7	**Task**: Learning the fundamental things at asp .net **Goal**: Set up database connections, build the infrastructure for a user login system, and create initial API endpoints. **Output**: A working asp .net project with a basic API structure and database connection.	Done

Zerda Gül: Learning ASP.NET	SOF-8	**Task**: Learning the fundamental things at asp .net **Goal**: Set up database connections, build the infrastructure for a user login system, and create initial API endpoints. **Output**: A working asp .net project with a basic API structure and database connection.	Done
Mehmet Emir Albayrakl: Learning ASP.NET	SOF-9	**Task**: Learning the fundamental things at asp .net **Goal**: Set up database connections, build the infrastructure for a user login system, and create initial API endpoints. **Output**: A working asp .net project with a basic API structure and database connection.	Done
Anhelina Bondarenko: Learning React and Figma Design	SOF-10	**Task**: Set up the React project and create basic components (navigation, header, footer, placeholder pages). **Goal**: Establish the skeleton structure of the application, set up page routing, and build the foundation for language selection. **Output**: A functional React project with placeholder pages and working language options. **Task**: Design the general layout, user interface, and interactions for all pages. **Goal**: Create mockups for both the user and admin interfaces and design a prototype for all screens. **Output**: A completed design document	Done

		in Figma, ready for further development.	

User and Database Management - Security

<table>
<tr><th>User Story</th><th>Key</th><th>Summary</th><th>Subtasks</th><th>Status</th></tr>
<tr>
<td>As a customer,I want to be able to create an account so that I can access the platform with my own credentials.</td>
<td>SOF-12</td>
<td>When a customer visits the platform, they should be able to create a new account by providing necessary details such as their name, email address, and password. The system should validate the input, securely store the information, and send a confirmation message if the registration is successful. This feature will ensure that users can have personalized access to the platform and maintain their own accounts.</td>
<td>
<table>
<tr><th>Summary</th><th>Key</th><th>Assignee</th><th>Status</th></tr>
<tr><td>Design Registration Form UI</td><td>SOF-67</td><td>Anhelina</td><td>Done</td></tr>
<tr><td>Create Submit Button</td><td>SOF-68</td><td>Anhelina</td><td>Done</td></tr>
<tr><td>Secure Password Input</td><td>SOF-60</td><td>Anhelina</td><td>Done</td></tr>
<tr><td>Integrate with Backend API</td><td>SOF-61</td><td>Anhelina</td><td>Done</td></tr>
<tr><td>User-Side Validation</td><td>SOF-59</td><td>Anhelina</td><td>Done</td></tr>
<tr><td>Database Integration</td><td>SOF-65</td><td>Zerda Gül</td><td>Done</td></tr>
<tr><td>Create User Registration API Endpoint</td><td>SOF-63</td><td>Zerda Gül</td><td>In Progress</td></tr>
<tr><td>Check for Duplicate Accounts</td><td>SOF-66</td><td>Zerda Gül</td><td>In Progress</td></tr>
</table>
</td>
<td>In Progress</td>
</tr>
</table>

<table>
<tr>
<td></td>
<td></td>
<td></td>
<td>
<table>
<tr><td>Input Validation & Sanitization</td><td>SOF-64</td><td>Zerda Gül</td><td>In Progress</td></tr>
<tr><td>Design Database schema</td><td>SOF-68</td><td>Zerda Gül</td><td>In Progress</td></tr>
<tr><td>Send Account Verification Email</td><td>SOF-67</td><td>Zerda Gül</td><td>To Do</td></tr>
</table>
</td>
<td></td>
</tr>
<tr>
<td>As a customer, I want to deactivate or delete my account so that I can remove my data from the platform.</td>
<td>SOF-13</td>
<td>Customers should have the ability to either temporarily deactivate or permanently delete their accounts. Deactivating an account will prevent the user from logging in until reactivated, while deleting an account will permanently remove the user's data from the platform. The process should be straightforward, secure, and should require confirmation to avoid accidental actions. Both options should ensure that user privacy is respected and data is handled securely.</td>
<td>
<table>
<tr><th>Summary</th><th>Key</th><th>Assignee</th><th>Status</th></tr>
<tr><td>Create Account Deletion API Endpoint</td><td>SOF-75</td><td>Zerda Gül</td><td>In Progress</td></tr>
<tr><td>Handle Data Removal or Archiving</td><td>SOF-76</td><td>Zerda Gül</td><td>In Progress</td></tr>
<tr><td>Database Updates</td><td>SOF-78</td><td>Zerda Gül</td><td>In Progress</td></tr>
<tr><td>Security Checks</td><td>SOF-77</td><td>Zerda Gül</td><td>In Progress</td></tr>
<tr><td>Create Database using SQL</td><td>SOF-79</td><td>Semih Sarıkoca</td><td>Done</td></tr>
<tr><td>Email Notification (optional)</td><td>SOF-89</td><td>Zerda Gül</td><td>In Progress</td></tr>
<tr><td>Create a menu in user profile page</td><td>SOF-167</td><td>Anhelina</td><td>Done</td></tr>
<tr><td>Integrate with Backend API</td><td>SOF-72</td><td>Anhelina</td><td>Done</td></tr>
<tr><td>Design Account Management UI</td><td>SOF-69</td><td>Anhelina</td><td>Done</td></tr>
<tr><td>Redirect After</td><td>SOF-</td><td>Anhelina</td><td>Done</td></tr>
</table>
</td>
<td>In Progress</td>
</tr>
</table>

<table>
<tr>
<td></td>
<td></td>
<td></td>
<td>
<table>
<tr><td>Account Deletion</td><td>73</td><td></td><td></td></tr>
<tr><td>Create Confirmation Page</td><td>SOF-70</td><td>Anhelina</td><td>Done</td></tr>
<tr><td>Form Validation and User Feedback (modal)</td><td>SOF-71</td><td>Anhelina</td><td>Done</td></tr>
</table>
</td>
<td></td>
</tr>
<tr>
<td>As a customer, I want to securely log in using my email and password so that only I can access my account.</td>
<td>SOF-14</td>
<td>This user story focuses on implementing a secure login system that allows customers to access their accounts using their email and password. The login process must ensure that only authenticated users can access their data and account functionalities. The system should handle common security concerns, such as hashing passwords., and providing feedback on incorrect login attempts.</td>
<td>
<table>
<tr><th>Summary</th><th>Key</th><th>Assignee</th><th>Status</th></tr>
<tr><td>Input Validation for Email and Password</td><td>SOF-82</td><td>Dicle Çoban</td><td>Done</td></tr>
<tr><td>Form Submission & API Integration</td><td>SOF-84</td><td>Dicle Çoban</td><td>Done</td></tr>
<tr><td>Design Login Page UI</td><td>SOF-81</td><td>Dicle Çoban</td><td>Done</td></tr>
<tr><td>Redirect After Successful Login</td><td>SOF-87</td><td>Dicle Çoban</td><td>Done</td></tr>
<tr><td>Define Database Relationships</td><td>SOF-83</td><td>Zerda Gül</td><td>To Do</td></tr>
<tr><td>Database Backup and Security Measures</td><td>SOF-85</td><td>Mehmet Emir Albayrak</td><td>To Do</td></tr>
<tr><td>Store User Session/Token (optional)</td><td>SOF-86</td><td>Semih Sarıkoca</td><td>Done</td></tr>
<tr><td>Create Login API Endpoint</td><td>SOF-88</td><td>Semih Sarıkoca</td><td>Done</td></tr>
<tr><td>Secure Password Storage (optional but recommended)</td><td>SOF-89</td><td>Semih Sarıkoca</td><td>Done</td></tr>
<tr><td>Session Management and</td><td>SOF-93</td><td>Semih Sarıkoca</td><td>Done</td></tr>
</table>
</td>
<td>Done</td>
</tr>
</table>

<table>
<tr><td></td><td></td><td></td><td>
<table>
<tr><td>Expiration (Optional)</td><td></td><td></td><td></td></tr>
<tr><td>Authenticate User Credentials</td><td>SOF-90</td><td>Semih Sarıkoca</td><td>Done</td></tr>
<tr><td>Rate Limiting and Brute-Force Protection (optional)</td><td>SOF-92</td><td>Semih Sarıkoca</td><td>Done</td></tr>
<tr><td>Generate Authentication Token (jwt token)</td><td>SOF-91</td><td>Semih Sarıkoca</td><td>Done</td></tr>
</table>
</td><td></td></tr>
<tr><td>As a customer, I want to reset my password if I forget it so that I can regain access to my account.</td><td>SOF-15</td><td>This user story involves creating a secure password reset process that enables customers to recover access to their accounts if they forget their password. The process should include requesting a password reset, verifying the customer's identity (via email verification or token), and allowing them to set a new password securely. The system must ensure that the process is secure, user-friendly, and prevents unauthorized access.</td><td>
<table>
<tr><th>Summary</th><th>Key</th><th>Assignee</th><th>Status</th></tr>
<tr><td>Design 'Forgot Password' Page UI</td><td>SOF-96</td><td>Dicle Çoban</td><td>Done</td></tr>
<tr><td>Create Password Reset Email Notification UI</td><td>SOF-98</td><td>Anhelina</td><td>Done</td></tr>
<tr><td>Handle Form Submission & API Integration</td><td>SOF-101</td><td>Anhelina</td><td>Done</td></tr>
<tr><td>Design Password Reset Form</td><td>SOF-99</td><td>Anhelina</td><td>Done</td></tr>
<tr><td>Input Validation for Email Address</td><td>SOF-97</td><td>Anhelina</td><td>Done</td></tr>
<tr><td>Loading Indicators and Feedback (modal)</td><td>SOF-102</td><td>Anhelina</td><td>Done</td></tr>
<tr><td>Form Validation for New Password</td><td>SOF-100</td><td>Anhelina</td><td>Done</td></tr>
<tr><td>Security Enhancements for Frontend (optional)</td><td>SOF-103</td><td>Mehmet Emir Albayrak</td><td>Done</td></tr>
</table>
</td><td>Done</td></tr>
</table>

<table>
<tr>
<td></td>
<td></td>
<td></td>
<td>
<table>
<tr><td>Create 'Forgot Password' API Endpoint</td><td>SOF-104</td><td>Mehmet Emir Albayrak</td><td>Done</td></tr>
<tr><td>Generate and Send Password Reset Email</td><td>SOF-105</td><td>Mehmet Emir Albayrak</td><td>Done</td></tr>
<tr><td>Create 'Reset Password' API Endpoint</td><td>SOF-106</td><td>Mehmet Emir Albayrak</td><td>Done</td></tr>
<tr><td>Token Security and Expiration (optional)</td><td>SOF-107</td><td>Mehmet Emir Albayrak</td><td>Done</td></tr>
<tr><td>Handle Password Reset Request</td><td>SOF-108</td><td>Mehmet Emir Albayrak</td><td>Done</td></tr>
<tr><td>Password Hashing (optional)</td><td>SOF-109</td><td>Mehmet Emir Albayrak</td><td>Done</td></tr>
<tr><td>Security Logging and Monitoring</td><td>SOF-111</td><td>Mehmet Emir Albayrak</td><td>Done</td></tr>
<tr><td>Encryption and Secure Communication (optional)</td><td>SOF-112</td><td>Mehmet Emir Albayrak</td><td>Done</td></tr>
</table>
</td>
<td></td>
</tr>
<tr>
<td>As a customer, I want to update my profile information (e.g., name, email, password) so that I can keep my account up-to-date.</td>
<td>SOF-16</td>
<td>This user story involves creating a user-friendly interface and backend functionality that allows customers to update their profile information, including their name, email, and password. The process should ensure that the updated information is validated and securely saved in the database.</td>
<td>
<table>
<tr><th>Summary</th><th>Key</th><th>Assignee</th><th>Status</th></tr>
<tr><td>Security Measures for Frontend</td><td>SOF-125</td><td>Dicle Çoban</td><td>In Progress</td></tr>
<tr><td>Responsive Design</td><td>SOF-120</td><td>Dicle Çoban</td><td>In Progress</td></tr>
<tr><td>Loading Indicators and Feedback</td><td>SOF-117</td><td>Dicle Çoban</td><td>Done</td></tr>
</table>
</td>
<td>In Progress</td>
</tr>
</table>

<table>
<tr>
<td></td>
<td></td>
<td>Additionally, the system must provide feedback to the user upon successful updates or errors encountered during the process.</td>
<td>
<table>
<tr><td>Messages</td><td></td><td></td><td></td></tr>
<tr><td>Implement Password Change Functionality</td><td>SOF-115</td><td>Dicle Çoban</td><td>Done</td></tr>
<tr><td>Handle Form Submission & API Integration</td><td>SOF-118</td><td>Dicle Çoban</td><td>Done</td></tr>
<tr><td>Input Validation for Profile Fields</td><td>SOF-122</td><td>Dicle Çoban</td><td>Done</td></tr>
<tr><td>Create a Submit Button</td><td>SOF-116</td><td>Dicle Çoban</td><td>Done</td></tr>
<tr><td>Design Profile Update Page UI</td><td>SOF-113</td><td>Dicle Çoban</td><td>Done</td></tr>
<tr><td>Create Profile Update API Endpoint</td><td>SOF-121</td><td>Mehmet Emir Albayrak</td><td>To Do</td></tr>
<tr><td>Input Validation on the Backend</td><td>SOF-122</td><td>Mehmet Emir Albayrak</td><td>To Do</td></tr>
<tr><td>Verify Current Password (if changing)</td><td>SOF-123</td><td>Mehmet Emir Albayrak</td><td>To Do</td></tr>
<tr><td>Update Profile Information in the Database</td><td>SOF-124</td><td>Mehmet Emir Albayrak</td><td>To Do</td></tr>
<tr><td>Security Measures for Backend</td><td>SOF-125</td><td>Mehmet Emir Albayrak</td><td>To Do</td></tr>
<tr><td>Error Handling</td><td>SOF-126</td><td>Mehmet Emir Albayrak</td><td>To Do</td></tr>
<tr><td>Create Navigation Bar</td><td>SOF-150</td><td>Anhelina</td><td>Done</td></tr>
</table>
</td>
<td></td>
</tr>
</table>

<table>
<tr><td>As a customer, I want to securely log out so that I can access my account another time.</td><td>SOF-142</td><td>This user story focuses on implementing a secure logout feature that ensures user sessions are properly terminated. When a user logs out, their session should be invalidated, and any sensitive data should be cleared from the client side. This feature will enhance security by preventing unauthorized access to the user's account after they leave the application.</td><td><table>
<tr><th>Summary</th><th>Key</th><th>Assignee</th><th>Status</th></tr>
<tr><td>Create Logout Button</td><td>SOF-143</td><td>Dicle Çoban</td><td>Done</td></tr>
<tr><td>Implement Logout Functionality (Integration)</td><td>SOF-144</td><td>Anhelina</td><td>In Progress</td></tr>
<tr><td>Handle Logout Response (optional)</td><td>SOF-146</td><td>Dicle Çoban</td><td>In Progress</td></tr>
<tr><td>Display Confirmation Modal</td><td>SOF-145</td><td>Dicle Çoban</td><td>In Progress</td></tr>
<tr><td>Create Logout Endpoint</td><td>SOF-147</td><td>Semih Sarıkoca</td><td>Done</td></tr>
<tr><td>Return Logout Confirmation Response (Optional)</td><td>SOF-149</td><td>Semih Sarıkoca</td><td>Done</td></tr>
</table></td><td>In Progress</td></tr>
</table>

Shopping Cart and Product Page

<table>
<tr><th>User Story</th><th>Key</th><th>Summary</th><th>Subtasks</th><th>Status</th></tr>
<tr><td>As a customer, I want to select products which I like, so I can easily buy them from my cart.</td><td>SOF-18</td><td>This user story focuses on allowing customers to select products they want to purchase, add them to their cart, and easily access them later for checkout. This involves implementing a seamless and user-friendly interface on the frontend to browse and select products, and</td><td><table>
<tr><th>Summary</th><th>Key</th><th>Assignee</th><th>Status</th></tr>
<tr><td>Design Product Listing Page UI</td><td>SOF-168</td><td></td><td>To Do</td></tr>
<tr><td>Add ‘Add to Cart’ Button and Icon</td><td>SOF-169</td><td></td><td>To Do</td></tr>
<tr><td>Product Detail Page Design</td><td>SOF-170</td><td></td><td>To Do</td></tr>
<tr><td>Handle Add to Cart</td><td>SOF-</td><td></td><td>To Do</td></tr>
</table></td><td>To Do</td></tr>
</table>

<table>
<tr>
<td></td>
<td></td>
<td>backend functionality to store and retrieve cart information.</td>
<td>
<table>
<tr><td>Action</td><td>172</td><td></td><td></td></tr>
<tr><td>Display Feedback Messages</td><td>SOF-173</td><td></td><td>To Do</td></tr>
<tr><td>Create Cart API Endpoint</td><td>SOF-174</td><td></td><td>To Do</td></tr>
<tr><td>Database Schema for Cart</td><td>SOF-175</td><td></td><td>To Do</td></tr>
<tr><td>Implement Stock Check on Add to Cart</td><td>SOF-176</td><td></td><td>To Do</td></tr>
<tr><td>Handle Stock Check on Add to Cart</td><td>SOF-177</td><td></td><td>To Do</td></tr>
<tr><td>Return Cart Summary</td><td>SOF-178</td><td></td><td>To Do</td></tr>
<tr><td>Session or Token Management for Guest Users (Optional)</td><td>SOF-179</td><td></td><td>To Do</td></tr>
<tr><td>Logging and Monitoring</td><td>SOF-180</td><td></td><td>To Do</td></tr>
</table>
</td>
<td></td>
</tr>
<tr>
<td>As a customer, I want to see my cart whenever I want, so I can check the products which I'll buy from my cart.</td>
<td>SOF-19</td>
<td>This user story focuses on providing customers with easy access to their cart so they can view the items they’ve selected for purchase. The feature should allow users to see a summary of their cart, including product names, quantities, prices, and the total cost. This requires frontend components to display the cart details and backend functionality to retrieve and manage the</td>
<td>
<table>
<tr><th>Summary</th><th>Key</th><th>Assignee</th><th>Status</th></tr>
<tr><td>Design Cart Page UI</td><td>SOF-181</td><td></td><td>To Do</td></tr>
<tr><td>Fetch Cart Data from Backend</td><td>SOF-182</td><td></td><td>To Do</td></tr>
<tr><td>Display Feedback Messages</td><td>SOF-183</td><td></td><td>To Do</td></tr>
<tr><td>Subtotal and Total Calculation</td><td>SOF-184</td><td></td><td>To Do</td></tr>
<tr><td>Add Cart Icon with Quick Access (Mostl Likely Need That)</td><td>SOF-185</td><td></td><td>To Do</td></tr>
</table>
</td>
<td>To Do</td>
</tr>
</table>

<table>
<tr>
<td></td>
<td></td>
<td>cart contents.</td>
<td>
<table>
<tr><td>Create Cart Retrieval API Endpoint</td><td>SOF-186</td><td></td><td>To Do</td></tr>
<tr><td>Implement Cart Summary Calculation</td><td>SOF-187</td><td></td><td>To Do</td></tr>
<tr><td>Error Handling</td><td>SOF-272</td><td></td><td>To Do</td></tr>
</table>
</td>
<td></td>
</tr>
<tr>
<td>As a customer, I want to manage amount of products which I put on my cart before, so I can buy everything according to my needs.</td>
<td>SOF-20</td>
<td>This user story involves allowing customers to adjust the quantity of products they have already added to their cart. The feature should be intuitive on the frontend, enabling users to easily increase or decrease the quantity of items. On the backend, quantity updates should be managed accurately, including recalculating the cart totals and checking product stock availability. This feature ensures customers can tailor their purchases to their requirements before proceeding to checkout.</td>
<td>
<table>
<tr><th>Summary</th><th>Key</th><th>Assignee</th><th>Status</th></tr>
<tr><td>Add Quantity Controls in Cart Page</td><td>SOF-188</td><td></td><td>To Do</td></tr>
<tr><td>Implement Quantity Input Validation</td><td>SOF-189</td><td></td><td>To Do</td></tr>
<tr><td>Update Cart Summary on Quantity Change</td><td>SOF-190</td><td></td><td>To Do</td></tr>
<tr><td>Error Handling and User Feedback (optional)</td><td>SOF-191</td><td></td><td>To Do</td></tr>
<tr><td>Create Quantity Update API Endpoint</td><td>SOF-192</td><td></td><td>To Do</td></tr>
<tr><td>Stock Validation on Quantity Change</td><td>SOF-193</td><td></td><td>To Do</td></tr>
<tr><td>Update Cart Totals</td><td>SOF-194</td><td></td><td>To Do</td></tr>
<tr><td>Logging Quantity Changes for Audit</td><td>SOF-195</td><td></td><td>To Do</td></tr>
</table>
</td>
<td>To Do</td>
</tr>
<tr>
<td>As a customer, I want to remove the producst which I don't</td>
<td>SOF-21</td>
<td>This user story involves giving customers the ability to remove items from their shopping cart,</td>
<td>
<table>
<tr><th>Summary</th><th>Key</th><th>Assignee</th><th>Status</th></tr>
<tr><td>Implement Remove Button in Cart UI</td><td>SOF-197</td><td></td><td>To Do</td></tr>
</table>
</td>
<td>To Do</td>
</tr>
</table>

<table>
<tr>
<td>want anymore, so with that just products that I wan't to buy remains in my cart.</td>
<td></td>
<td>ensuring they can keep their cart organized with only the products they intend to purchase. This requires both frontend and backend functionality, where users can initiate product removal, and the backend updates the cart accordingly.</td>
<td>
<table>
<tr><td>Handle Remove Action and Update Cart Display</td><td>SOF-198</td><td></td><td>To Do</td></tr>
<tr><td>Confirm Removal Action (Optional)</td><td>SOF-199</td><td></td><td>To Do</td></tr>
<tr><td>Update Cart Totals</td><td>SOF-200</td><td></td><td>To Do</td></tr>
<tr><td>Display Feedback Messages (optional)</td><td>SOF-201</td><td></td><td>To Do</td></tr>
<tr><td>Create Remove Item API Endpoint</td><td>SOF-202</td><td></td><td>To Do</td></tr>
<tr><td>Verify Item Ownership in Cart</td><td>SOF-203</td><td></td><td>To Do</td></tr>
<tr><td>Update Cart Database</td><td>SOF-204</td><td></td><td>To Do</td></tr>
<tr><td>Handle Errors and Validation</td><td>SOF-205</td><td></td><td>To Do</td></tr>
<tr><td>Log Removal Actions (Optional)</td><td>SOF-206</td><td></td><td>To Do</td></tr>
</table>
</td>
<td></td>
</tr>
<tr>
<td>As a customer, I want to view detailed information about a product so that I can make an informed purchasing decision.</td>
<td>SOF-23</td>
<td>This user story focuses on providing customers with comprehensive product details, helping them understand the product better before making a purchase. This feature includes displaying key information such as product name, price, description, availability, customer reviews, and any relevant specifications or options (e.g., size, color). By offering clear and thorough product</td>
<td>
<table>
<tr><th>Summary</th><th>Key</th><th>Assignee</th><th>Status</th></tr>
<tr><td>Design Product Detail Page UI</td><td>SOF-207</td><td></td><td>To Do</td></tr>
<tr><td>Display Product Images with Zoom Feature (optional)</td><td>SOF-208</td><td></td><td>To Do</td></tr>
<tr><td>Implement Product Specifications and Options</td><td>SOF-209</td><td></td><td>To Do</td></tr>
<tr><td>Add 'Add to Cart' Button</td><td>SOF-210</td><td></td><td>To Do</td></tr>
<tr><td>Customer Reviews and Ratings Section</td><td>SOF-211</td><td></td><td>To Do</td></tr>
</table>
</td>
<td>To Do</td>
</tr>
</table>

<table>
<tr>
<td></td>
<td></td>
<td>information, customers can make confident and informed buying decisions.</td>
<td>
<table>
<tr><td>Create Product Detail API Endpoint</td><td>SOF-212</td><td></td><td>To Do</td></tr>
<tr><td>Fetch and Display Customer Reviews</td><td>SOF-213</td><td></td><td>To Do</td></tr>
<tr><td>Handle Product Stock Availability</td><td>SOF-214</td><td></td><td>To Do</td></tr>
<tr><td>Logging and Monitoring</td><td>SOF-215</td><td></td><td>To Do</td></tr>
</table>
</td>
<td></td>
</tr>
<tr>
<td>As a customer, I want to browse products by category, so I can find the types of products I'm looking for.</td>
<td>SOF-35</td>
<td>This user story focuses on allowing customers to browse products by specific categories, making it easier for them to find items of interest quickly. The feature will display all product categories and let users filter and explore products within each one. It involves creating a user-friendly interface for selecting categories on the frontend and backend functionality for retrieving and displaying products based on category selection.</td>
<td>
<table>
<tr><th>Summary</th><th>Key</th><th>Assignee</th><th>Status</th></tr>
<tr><td>Design Product Sidebar Menu</td><td>SOF-241</td><td></td><td>To Do</td></tr>
<tr><td>Create Filter Bar with Tags</td><td>SOF-242</td><td></td><td>To Do</td></tr>
<tr><td>Add Sorting Dropdown</td><td>SOF-243</td><td></td><td>To Do</td></tr>
<tr><td>Design Product Card Component</td><td>SOF-244</td><td></td><td>To Do</td></tr>
<tr><td>Implement Pagination</td><td>SOF-245</td><td></td><td>To Do</td></tr>
<tr><td>Responsive Design Testing</td><td>SOF-246</td><td></td><td>To Do</td></tr>
<tr><td>API Endpoint for Product List</td><td>SOF-247</td><td></td><td>To Do</td></tr>
<tr><td>API Endpoint for Category List</td><td>SOF-248</td><td></td><td>To Do</td></tr>
<tr><td>Implement Filtering Logic by Tags</td><td>SOF-249</td><td></td><td>To Do</td></tr>
<tr><td>Sorting and Pagination Logic</td><td>SOF-250</td><td></td><td>To Do</td></tr>
</table>
</td>
<td>To Do</td>
</tr>
</table>

<table>
<tr>
<td></td>
<td></td>
<td></td>
<td>
<table>
<tr><td>Optimize Database Queries</td><td>SOF-251</td><td></td><td>To Do</td></tr>
</table>
</td>
<td></td>
</tr>
<tr>
<td>As a customer, I want to search for products by entering keywords in a search bar, so I can quickly find specific items.</td>
<td>SOF-37</td>
<td>This user story focuses on providing a search bar that allows customers to easily find specific products by entering keywords. The search function should quickly return relevant results based on product names, descriptions, or other attributes, helping customers locate what they need without extensive browsing. By enabling efficient and accurate searches, this feature improves the shopping experience and makes it more convenient for customers to find desired items.</td>
<td>
<table>
<tr><th>Summary</th><th>Key</th><th>Assignee</th><th>Status</th></tr>
<tr><td>Design Search Bar UI</td><td>SOF-252</td><td></td><td>To Do</td></tr>
<tr><td>Implement Real-Time Search Suggestions (Optional)</td><td>SOF-253</td><td></td><td>To Do</td></tr>
<tr><td>Build Search Results Page Layout</td><td>SOF-254</td><td></td><td>To Do</td></tr>
<tr><td>Clear Search Option</td><td>SOF-255</td><td></td><td>To Do</td></tr>
<tr><td>Create Search API Endpoint</td><td>SOF-256</td><td></td><td>To Do</td></tr>
<tr><td>Implement Keyword Matching Logic</td><td>SOF-257</td><td></td><td>To Do</td></tr>
<tr><td>Optimize Database Query for Search</td><td>SOF-258</td><td></td><td>To Do</td></tr>
<tr><td>Pagination and Limit Results</td><td>SOF-259</td><td></td><td>To Do</td></tr>
<tr><td>Log Search Queries and Monitor Performance</td><td>SOF-260</td><td></td><td>To Do</td></tr>
<tr><td>Handle Empty Results and Edge Cases</td><td>SOF-261</td><td></td><td>To Do</td></tr>
</table>
</td>
<td>To Do</td>
</tr>
</table>

Contact and Frequently Asked Questions

<table>
<tr><th>User Story</th><th>Key</th><th>Summary</th><th>Subtasks</th><th>Status</th></tr>
<tr><td>As a customer, I want the communication to be completed so that I can reach the Support Team</td><td>SOF-26</td><td>In this feature, we aim to enhance the communication experience for our customers by providing a dedicated section where they can easily reach out to our support team. The communication section will include a user-friendly contact form, allowing customers to submit their inquiries, feedback, or concerns directly. Overall, this feature will improve user satisfaction by streamlining the communication process.</td><td>
<table>
<tr><th>Summary</th><th>Key</th><th>Assignee</th><th>Status</th></tr>
<tr><td>Design Contact Form Section UI</td><td>SOF-39</td><td></td><td>To Do</td></tr>
<tr><td>Develop Contact Form</td><td>SOF-40</td><td></td><td>To Do</td></tr>
<tr><td>Test the Contact form.</td><td>SOF-41</td><td></td><td>To Do</td></tr>
<tr><td>API Endpoint for Submitting Support Requests</td><td>SOF-52</td><td></td><td>To Do</td></tr>
<tr><td>Database Schema Design for Support Requests</td><td>SOF-53</td><td></td><td>To Do</td></tr>
<tr><td>Automated Email Acknowledgment to Customers (optional)</td><td>SOF-54</td><td></td><td>To Do</td></tr>
</table>
</td><td>To Do</td></tr>
<tr><td>As a customer, I want to quickly find answers to my problems by reviewing the frequently asked questions page.</td><td>SOF-27</td><td>This feature aims to enhance the customer experience by providing a well-structured and easily accessible Frequently Asked Questions (FAQ) page. The FAQ page will serve as a self-service resource where customers can quickly find answers to common questions,</td><td>
<table>
<tr><th>Summary</th><th>Key</th><th>Assignee</th><th>Status</th></tr>
<tr><td>Design FAQ Page Layout</td><td>SOF-43</td><td></td><td>To Do</td></tr>
<tr><td>Develop FAQ Search Feature</td><td>SOF-44</td><td></td><td>To Do</td></tr>
<tr><td>Organize FAQ Content</td><td>SOF-45</td><td></td><td>To Do</td></tr>
<tr><td>Add FAQ Page Link to Navigation Bar</td><td>SOF-46</td><td></td><td>To Do</td></tr>
</table>
</td><td>To Do</td></tr>
</table>

<table>
<tr><td></td><td></td><td>saving time and reducing the need to contact customer support.

The page will be organized into categories, ensuring that users can navigate through different topics effortlessly. Additionally, a search functionality will be implemented to allow customers to quickly locate specific answers. By offering clear, concise, and accurate information, the FAQ page will help customers resolve their issues independently and efficiently.</td><td><table>
<tr><td>Test FAQ Page Usability</td><td>SOF-47</td><td></td><td>To Do</td></tr>
<tr><td>Implement Feedback Mechanism(optional)</td><td>SOF-48</td><td></td><td>To Do</td></tr>
<tr><td>Database Schema Design for FAQ</td><td>SOF-49</td><td></td><td>To Do</td></tr>
<tr><td>Backend Validation and Error Handling</td><td>SOF-50</td><td></td><td>To Do</td></tr>
<tr><td>Update Operations for FAQ Management</td><td>SOF-51</td><td></td><td>To Do</td></tr>
</table></td><td></td></tr>
</table>

Control Panel and Management

<table>
<tr><th>User Story</th><th>Key</th><th>Summary</th><th>Subtasks</th><th>Status</th></tr>
<tr><td>As a product admin, I want to add, edit, and delete products so that I can manage the catalog efficiently.</td><td>SOF-29</td><td>This user story enables the product admin to have full control over the product catalog by allowing them to add new products, edit existing product details, and delete products when necessary. This functionality ensures that the catalog is always up-to-date with accurate product information,</td><td><table>
<tr><th>Summary</th><th>Key</th><th>Assignee</th><th>Status</th></tr>
<tr><td>Adding the data into cloud</td><td>SOF-140</td><td></td><td>To Do</td></tr>
<tr><td>Design Product Management Dashboard</td><td>SOF-216</td><td></td><td>To Do</td></tr>
<tr><td>Build Add Product Form</td><td>SOF-217</td><td></td><td>To Do</td></tr>
</table></td><td>To Do</td></tr>
</table>

<table>
<tr>
<td></td>
<td></td>
<td>making it easier to manage inventory and meet customer expectations. A user-friendly interface and backend support for these actions streamline catalog management, saving time and reducing errors.</td>
<td>
<table>
<tr><td>Build Edit Product Form</td><td>SOF-218</td><td></td><td>To Do</td></tr>
<tr><td>Implement Delete Product Confirmation Dialog</td><td>SOF-219</td><td></td><td>To Do</td></tr>
<tr><td>Handle Product Image Upload and Preview (optional)</td><td>SOF-220</td><td></td><td>To Do</td></tr>
<tr><td>Create Add Product API Endpoint</td><td>SOF-221</td><td></td><td>To Do</td></tr>
<tr><td>Create Edit Product API Endpoint</td><td>SOF-222</td><td></td><td>To Do</td></tr>
<tr><td>Create Delete Product API Endpoint</td><td>SOF-223</td><td></td><td>To Do</td></tr>
<tr><td>Handle Image Storage and Retrieval</td><td>SOF-224</td><td></td><td>To Do</td></tr>
<tr><td>Error Handling</td><td>SOF-273</td><td></td><td>To Do</td></tr>
</table>
</td>
<td></td>
</tr>
<tr>
<td>As an order admin, I want to manage orders and track their status so that I can ensure timely processing and delivery to customers.</td>
<td>SOF-30</td>
<td>This user story is focused on providing the order admin with a comprehensive system to manage and track customer orders. The admin should be able to view all orders, update their status (e.g., "Requests," "In Progress" "Done"), and manage specific order details, ensuring each order progresses smoothly from initiation to delivery. This functionality requires both frontend and backend implementations to support efficient order</td>
<td>
<table>
<tr><th>Summary</th><th>Key</th><th>Assignee</th><th>Status</th></tr>
<tr><td>Design Order Management Dashboard</td><td>SOF-225</td><td></td><td>To Do</td></tr>
<tr><td>Implement Order Detail View</td><td>SOF-226</td><td></td><td>To Do</td></tr>
<tr><td>Add Order Status Update Controls (optional)</td><td>SOF-227</td><td></td><td>To Do</td></tr>
<tr><td>Display Real-Time Status Updates</td><td>SOF-228</td><td></td><td>To Do</td></tr>
<tr><td>User Feedback and Notifications (optional)</td><td>SOF-229</td><td></td><td>To Do</td></tr>
</table>
</td>
<td>To Do</td>
</tr>
</table>

<table>
<tr>
<td></td>
<td></td>
<td>tracking and status management.</td>
<td>
<table>
<tr><td>Create Order Management API Endpoint</td><td>SOF-230</td><td></td><td>To Do</td></tr>
<tr><td>Implement Order Detail Retrieval Endpoint</td><td>SOF-231</td><td></td><td>To Do</td></tr>
<tr><td>Create Order Status Update API Endpoint</td><td>SOF-232</td><td></td><td>To Do</td></tr>
<tr><td>Error Handling</td><td>SOF-275</td><td></td><td>To Do</td></tr>
</table>
</td>
<td></td>
</tr>
<tr>
<td>As a product admin, I want to track stock levels and receive notifications when stock is low, so I can restock in time.</td>
<td>SOF-31</td>
<td>This user story focuses on enabling the product admin to monitor stock levels efficiently and receive timely notifications when stock for any product reaches a low threshold. This helps ensure products are available for customers without interruption. The frontend provides a clear display of stock levels and notification settings, while the backend manages stock tracking and notification triggers.</td>
<td>
<table>
<tr><th>Summary</th><th>Key</th><th>Assignee</th><th>Status</th></tr>
<tr><td>Display Stock Levels in Product Management Dashboard</td><td>SOF-262</td><td></td><td>To Do</td></tr>
<tr><td>Create Low Stock Notification Settings</td><td>SOF-263</td><td></td><td>To Do</td></tr>
<tr><td>Low Stock Notification Display</td><td>SOF-264</td><td></td><td>To Do</td></tr>
<tr><td>Filter and Sort by Stock Levels</td><td>SOF-265</td><td></td><td>To Do</td></tr>
<tr><td>Develop Stock Monitoring System</td><td>SOF-266</td><td></td><td>To Do</td></tr>
<tr><td>Configure Low Stock Thresholds in Database</td><td>SOF-267</td><td></td><td>To Do</td></tr>
<tr><td>Implement Low Stock Notification Logic</td><td>SOF-268</td><td></td><td>To Do</td></tr>
<tr><td>Notification API Endpoint</td><td>SOF-269</td><td></td><td>To Do</td></tr>
</table>
</td>
<td>To Do</td>
</tr>
</table>

<table>
<tr><td></td><td></td><td></td><td><table>
<tr><td>Manage Notification Status</td><td>SOF-270</td><td></td><td>To Do</td></tr>
<tr><td>Error Handling</td><td>SOF-274</td><td></td><td>To Do</td></tr>
</table></td><td></td></tr>
<tr><td>As an order admin, I want to see the stocks of the products in Dashboard.</td><td>SOF-271</td><td>This user story is focused on providing the order admin with visibility into product stock levels directly within the dashboard. This feature will help the order admin keep track of inventory while processing and managing orders, ensuring that products are available and allowing proactive steps if stocks are low. The dashboard should display stock levels for each product in a clear and organized manner.</td><td><table>
<tr><th>Summary</th><th>Key</th><th>Assignee</th><th>Status</th></tr>
<tr><td>Design Stock Display in Order Management Dashboard</td><td>SOF-276</td><td></td><td>To Do</td></tr>
<tr><td>Implement Low Stock Indicators</td><td>SOF-277</td><td></td><td>To Do</td></tr>
<tr><td>Add Sorting and Filtering Options for Stock Levels (optional)</td><td>SOF-278</td><td></td><td>To Do</td></tr>
<tr><td>Tooltip or Detail View for Additional Stock Information (optional)</td><td>SOF-279</td><td></td><td>To Do</td></tr>
<tr><td>Responsive Design for Dashboard View</td><td>SOF-280</td><td></td><td>To Do</td></tr>
<tr><td>Create API Endpoint for Product Stock Levels</td><td>SOF-281</td><td></td><td>To Do</td></tr>
<tr><td>Optimize Database Query for Stock Retrieval</td><td>SOF-282</td><td></td><td>To Do</td></tr>
<tr><td>Configure Low Stock Thresholds</td><td>SOF-283</td><td></td><td>To Do</td></tr>
<tr><td>Data Caching for Frequent Access (Optional)</td><td>SOF-284</td><td></td><td>To Do</td></tr>
<tr><td>Error Handling and Validation</td><td>SOF-285</td><td></td><td>To Do</td></tr>
</table></td><td>To Do</td></tr>
</table>

Home Page and About Us

User Story	Key	Summary	Subtasks				Status
			Summary	**Key**	**Assignee**	**Status**	
As a customer, I want to see a nice order homepage that showcases featured products, categories, and promotional banners, so I can quickly find items of interest.	SOF-34	This user story focuses on creating an attractive and well-organized homepage for customers, featuring a visually engaging layout that highlights popular products, key categories, and current promotions. The homepage is designed to make it easy for customers to discover items of interest at a glance, guiding them toward products and deals that may appeal to them. By providing clear navigation and visually distinct sections, the homepage enhances the shopping experience, encouraging exploration and helping customers quickly find what they are looking for.	Design Homepage Layout	SOF-233		To Do	To Do
			Implement Featured Products Section	SOF-234		To Do	
			Create Promotional Banner Slider (optional)	SOF-235		To Do	
			Add Categories Section	SOF-236		To Do	
			Loading Indicators and Error Messages	SOF-237		To Do	
			Create API Endpoint for Featured Products	SOF-238		To Do	
			Develop API Endpoint for Promotional Banners	SOF-239		To Do	
			API Endpoint for Categories	SOF-240		To Do	

UML (Unified Modeling Language) Diagrams of the Project

a. Use Case Diagram of Eko Inovation Website

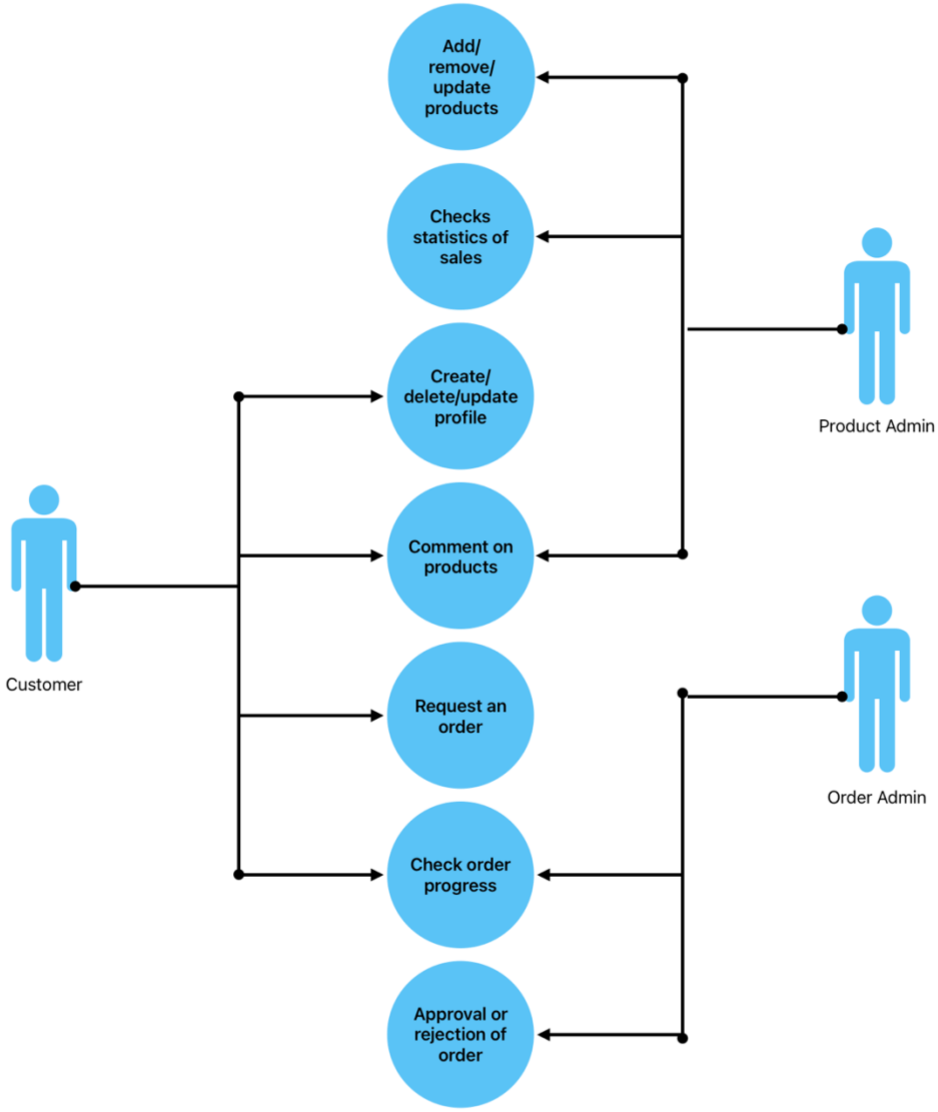

Add/ remove/ update products
Checks statistics of sales
Create/ delete/update profile
Comment on products
Request an order
Check order progress
Approval or rejection of order
Customer
Product Admin
Order Admin

Figure a.1 Use cases

b. Use Case by Roles

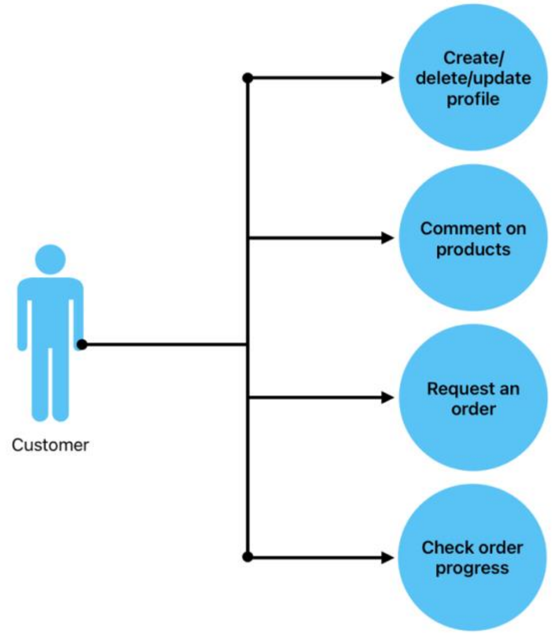


Figure b.1 Use cases involving the role “Customer”

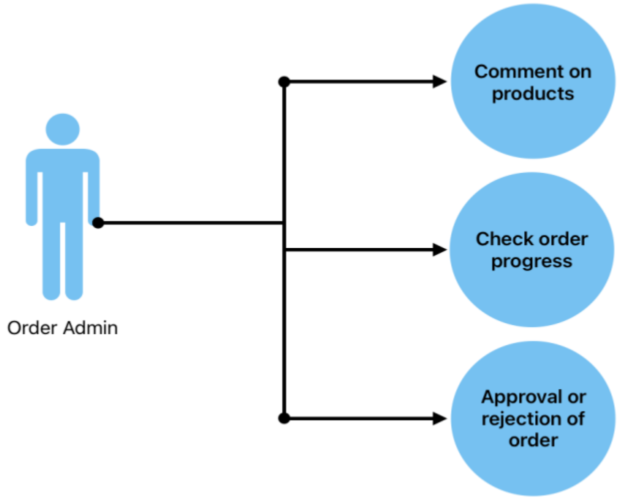


Figure b.2 Use cases involving the role “Order Admin”

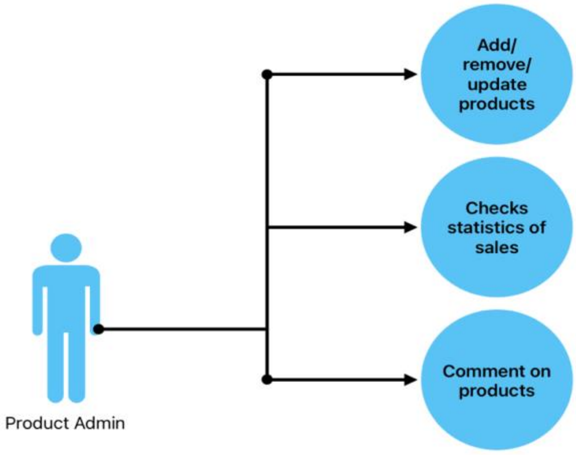


Figure b.3 Use cases involving the role “Product Admin”

Use Case Tables

Use Case Table: Add/Remove/Update Products	
Actors	Product Admin
Description	The Product Admin can add new products, update existing products, or remove products from the system.
Data	Product name, description, price, stock information
Stimulus	The Product Admin initiates the action to add, update, or remove a product.
Response	Product information is updated in the database, and the product list and stock status are refreshed.
Comments	This feature enables the Product Admin to manage the product catalog and stock availability.

Use Case Table: Check Statistics of Sales	
Actors	Product Admin
Description	The Product Admin views sales statistics to analyze sales performance.
Data	Sales data, number of units sold per product, total revenue
Stimulus	The Product Admin accesses the sales statistics dashboard.
Response	Sales data is analyzed and displayed to the Product Admin.
Comments	This feature allows the Product Admin to examine sales trends and performance insights.

Use Case Table: Create/Delete/Update Profile	
Actors	Customer
Description	The Customer can create, update, or delete their profile.
Data	Customer information (name, email, address, etc.)
Stimulus	The Customer creates, updates, or deletes their profile.
Response	Profile information is saved, updated, or removed from the database.
Comments	This feature allows customers to keep their profile information up-to-date for streamlined order processing.

Use Case Table: Comment on Products	
Actors	Customer, Order Admin
Description	The Customer and Order Admin can communicate about products through comments.
Data	Comment text, customer information, product information
Stimulus	The Customer leaves a comment or question about a product, and the Order Admin may respond with relevant information.
Response	The comment is saved in the database and displayed on the product page.
Comments	This feature allows customers to inquire about products and receive responses from the Order Admin.

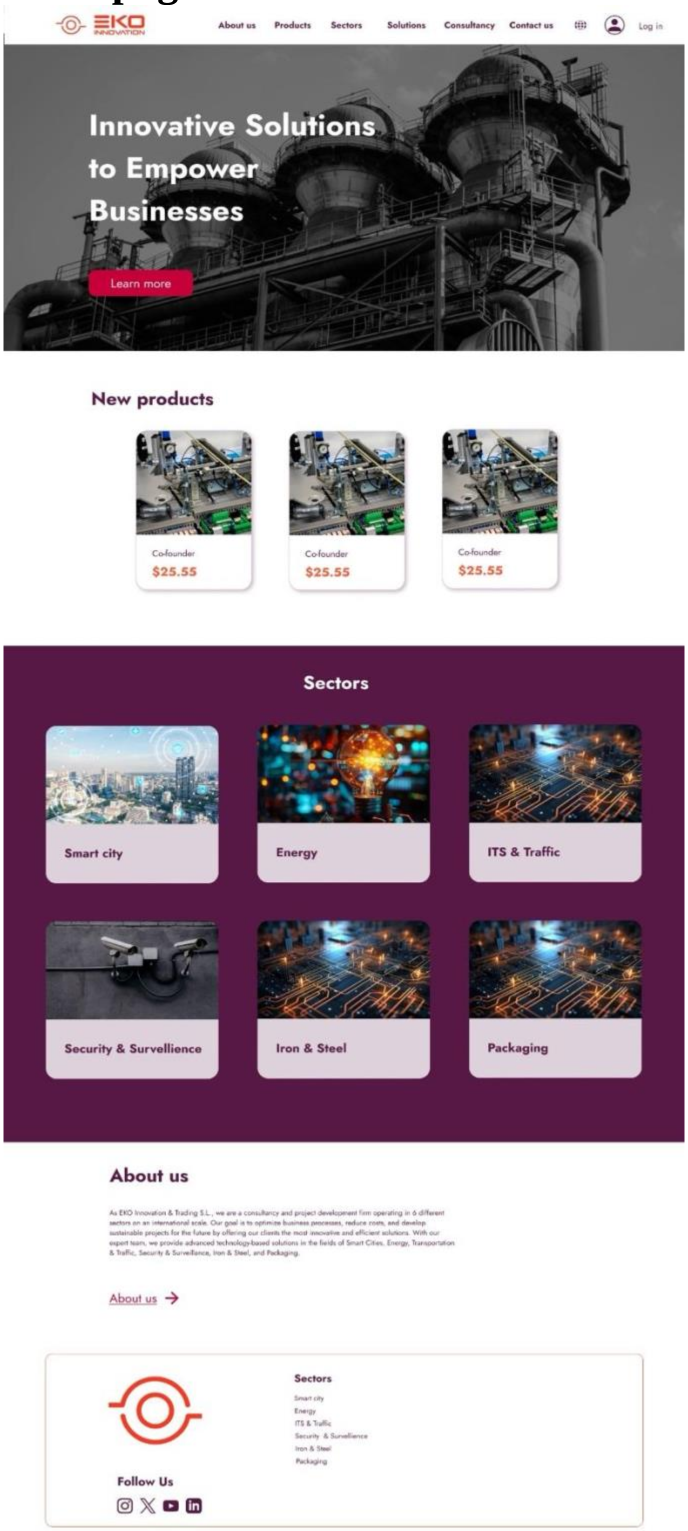
Use Case Table: Request an Order	
Actors	Customer
Description	The Customer can place an order request.
Data	Order information (product, quantity, payment details, etc.)
Stimulus	The Customer adds products to the cart and completes the checkout process.
Response	The order is saved in the database, and order processing begins.
Comments	This feature allows customers to place orders for products.

Use Case Table: Check Order Progress	
Actors	Customer
Description	The Customer checks the status of their current orders.
Data	Order status (confirmed, processing, shipped, etc.)
Stimulus	The Customer accesses the order status screen.
Response	The order status is displayed on the screen.
Comments	This feature provides customers with updates on the progress of their orders.

Use Case Table: Approval or Rejection of Order	
Actors	Order Admin
Description	The Order Admin reviews and either approves or rejects incoming orders.
Data	Order information, customer information
Stimulus	The Order Admin reviews the order and decides to approve or reject it.
Response	The order status is updated, and the Customer is notified accordingly.
Comments	This feature allows the Order Admin to manage order approvals and keep customers informed.

User Interface

Main page



Innovative Solutions to Empower Businesses

New products

Co-founder $25.55	Co-founder $25.55	Co-founder $25.55

Sectors

Smart city	Energy	ITS & Traffic
Security & Survellience	Iron & Steel	Packaging

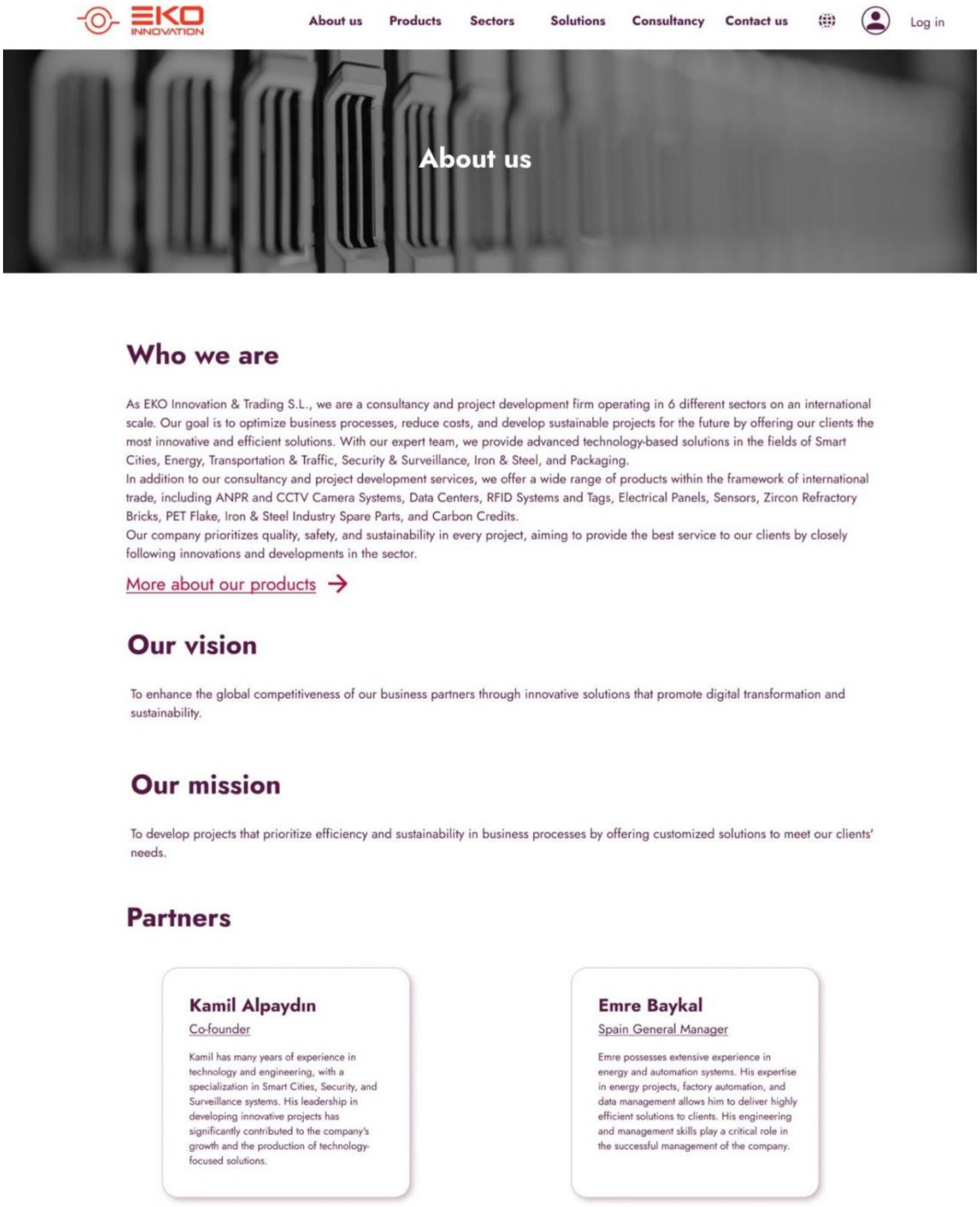
About us

As EKO Innovation & Trading S.L., we are a consultancy and project development firm operating in 6 different sectors on an international scale. Our goal is to optimize business processes, reduce costs, and develop sustainable projects for the future by offering our clients the most innovative and efficient solutions. With our expert team, we provide advanced technology-based solutions in the fields of Smart Cities, Energy, Transportation & Traffic, Security & Surveillance, Iron & Steel, and Packaging.

Sectors

Smart city
Energy
ITS & Traffic
Security & Survellience
Iron & Steel
Packaging

About us



Who we are

As EKO Innovation & Trading S.L., we are a consultancy and project development firm operating in 6 different sectors on an international scale. Our goal is to optimize business processes, reduce costs, and develop sustainable projects for the future by offering our clients the most innovative and efficient solutions. With our expert team, we provide advanced technology-based solutions in the fields of Smart Cities, Energy, Transportation & Traffic, Security & Surveillance, Iron & Steel, and Packaging.

In addition to our consultancy and project development services, we offer a wide range of products within the framework of international trade, including ANPR and CCTV Camera Systems, Data Centers, RFID Systems and Tags, Electrical Panels, Sensors, Zircon Refractory Bricks, PET Flake, Iron & Steel Industry Spare Parts, and Carbon Credits.

Our company prioritizes quality, safety, and sustainability in every project, aiming to provide the best service to our clients by closely following innovations and developments in the sector.

More about our products →

Our vision

To enhance the global competitiveness of our business partners through innovative solutions that promote digital transformation and sustainability.

Our mission

To develop projects that prioritize efficiency and sustainability in business processes by offering customized solutions to meet our clients' needs.

Partners

Kamil Alpaydın

Co-founder

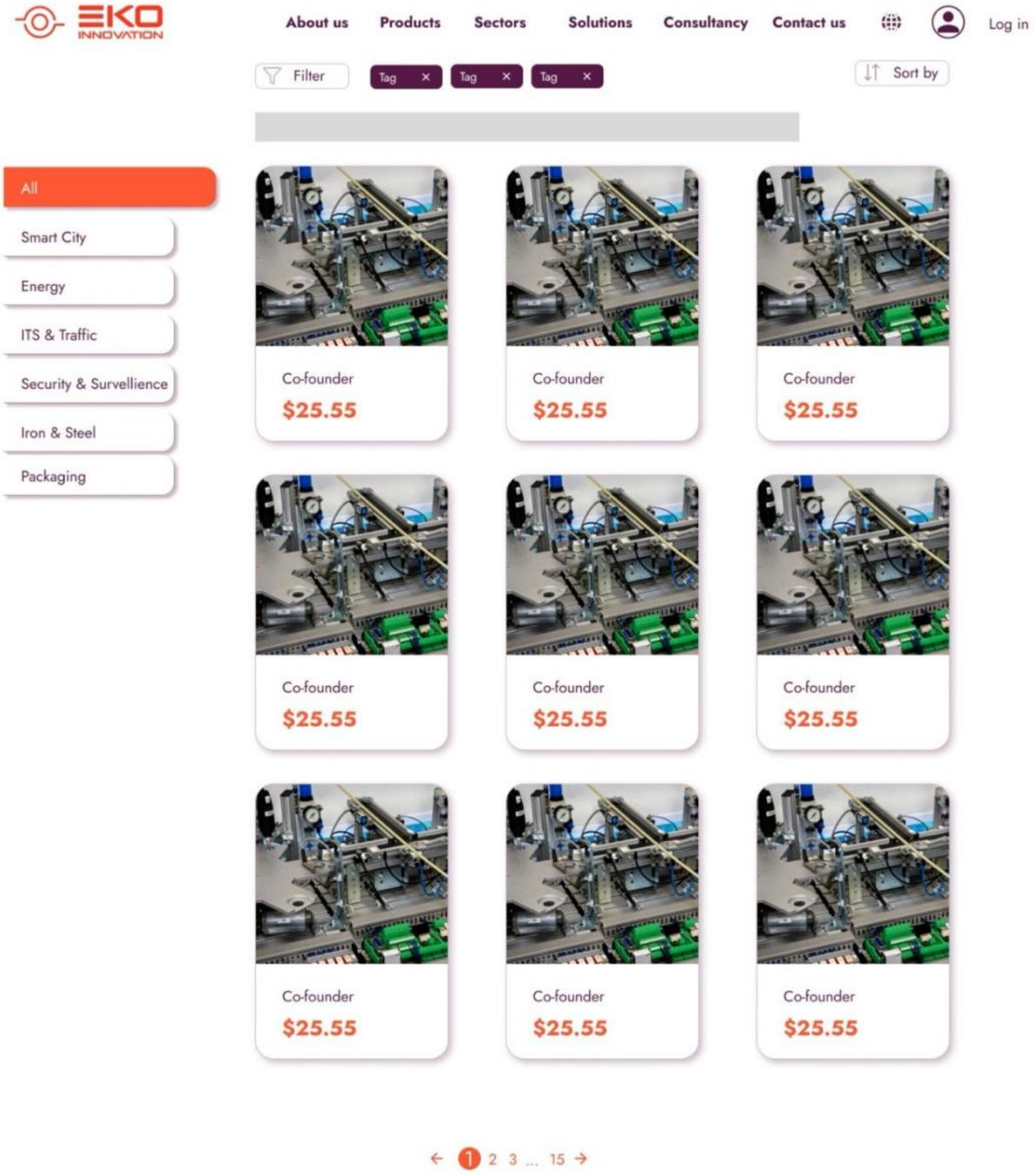
Kamil has many years of experience in technology and engineering, with a specialization in Smart Cities, Security, and Surveillance systems. His leadership in developing innovative projects has significantly contributed to the company's growth and the production of technology-focused solutions.

Emre Baykal

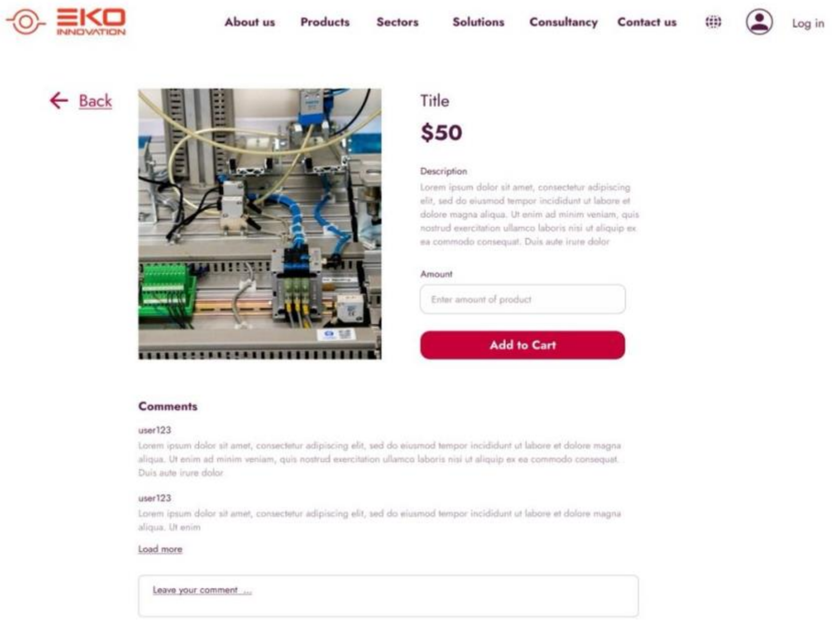
Spain General Manager

Emre possesses extensive experience in energy and automation systems. His expertise in energy projects, factory automation, and data management allows him to deliver highly efficient solutions to clients. His engineering and management skills play a critical role in the successful management of the company.

Products


EKO INNOVATION
About us
Products
Sectors
Solutions
Consultancy
Contact us
Log in
Filter
Tag ×
Tag ×
Tag ×
Sort by
All
Smart City
Energy
ITS & Traffic
Security & Survellience
Iron & Steel
Packaging
Co-founder
$25.55
Co-founder
$25.55
Co-founder
$25.55
Co-founder
$25.55
Co-founder
$25.55
Co-founder
$25.55
Co-founder
$25.55
Co-founder
$25.55
Co-founder
$25.55
← 1 2 3 ... 15 →

Product page



Title

$50

Description

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor

Amount: Enter amount of product

Comments

user123

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor

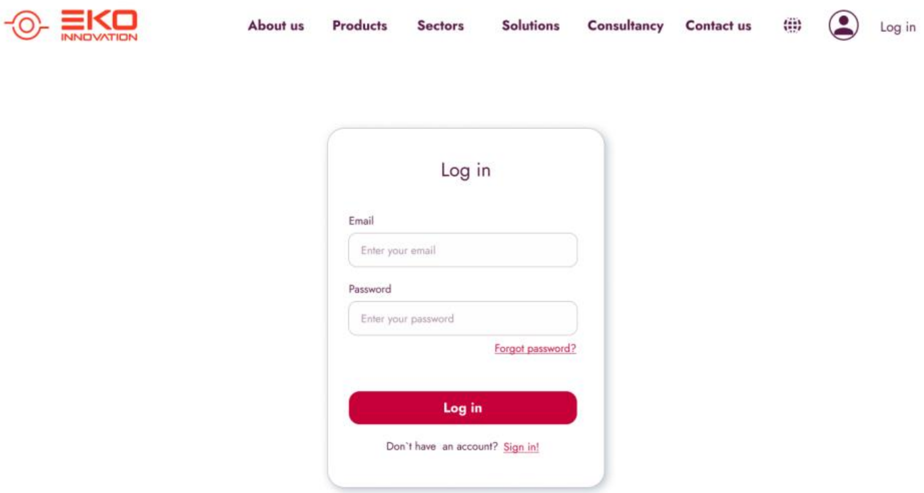
user123

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim

Leave your comment ...

User management

Log in



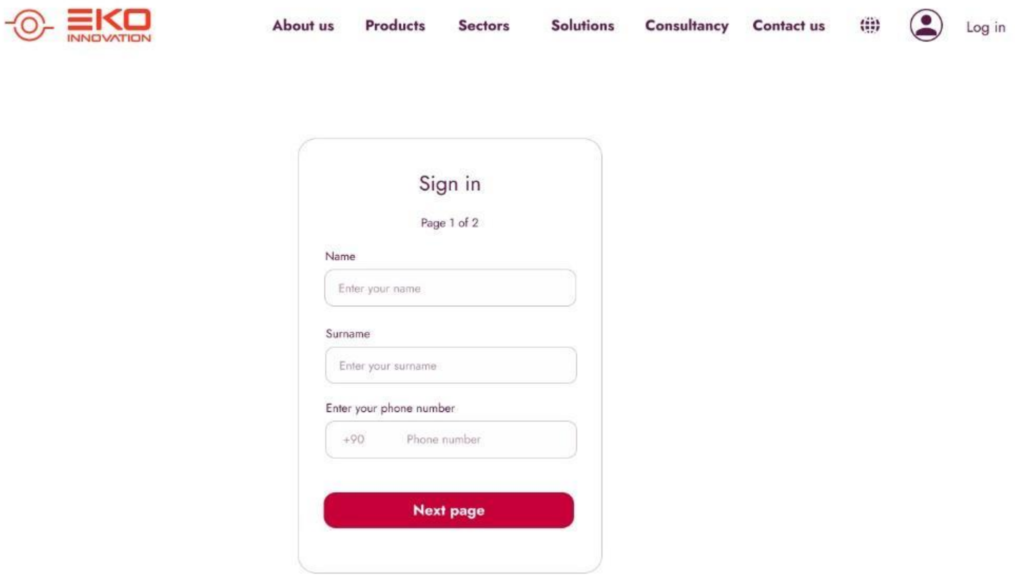
Log in

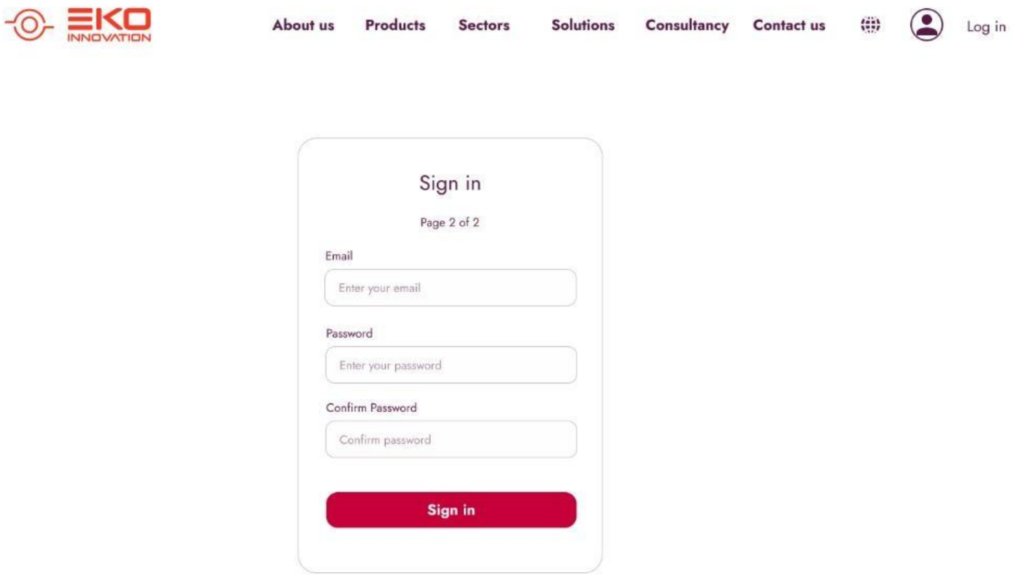
Email: Enter your email

Password: Enter your password

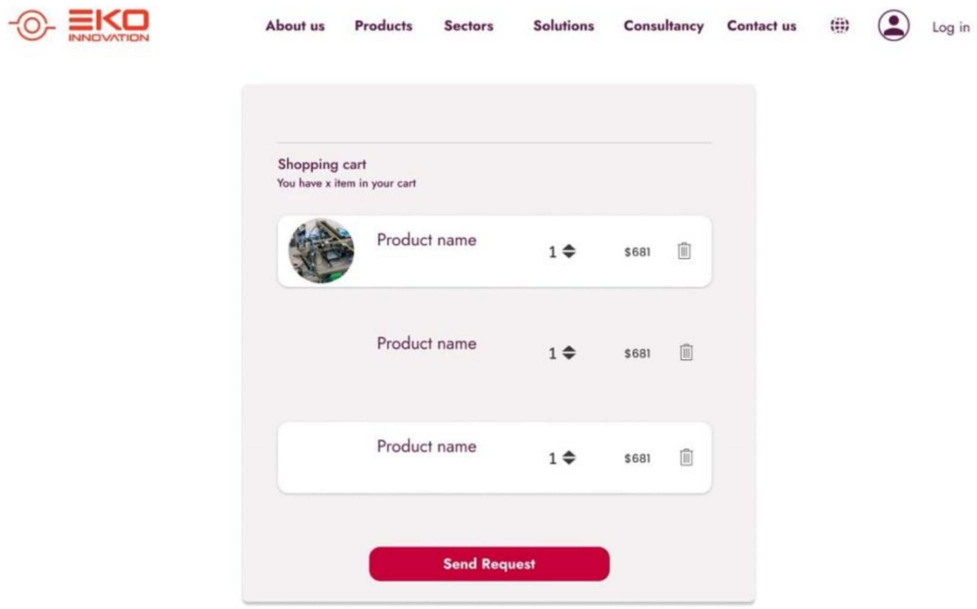
Don't have an account? Sign in!

Registration





Shopping cart (accessed from navigation panel)



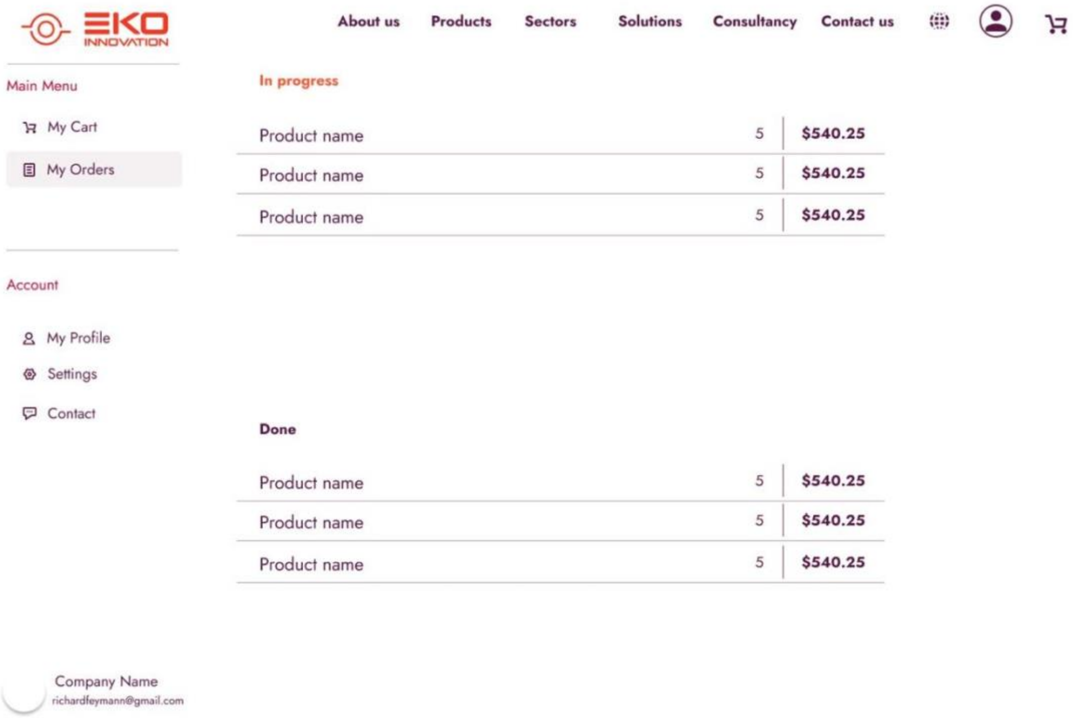
Shopping cart

You have x item in your cart

Product name	1	$681
Product name	1	$681
Product name	1	$681

Profile Panel for User

Orders history and status



In progress

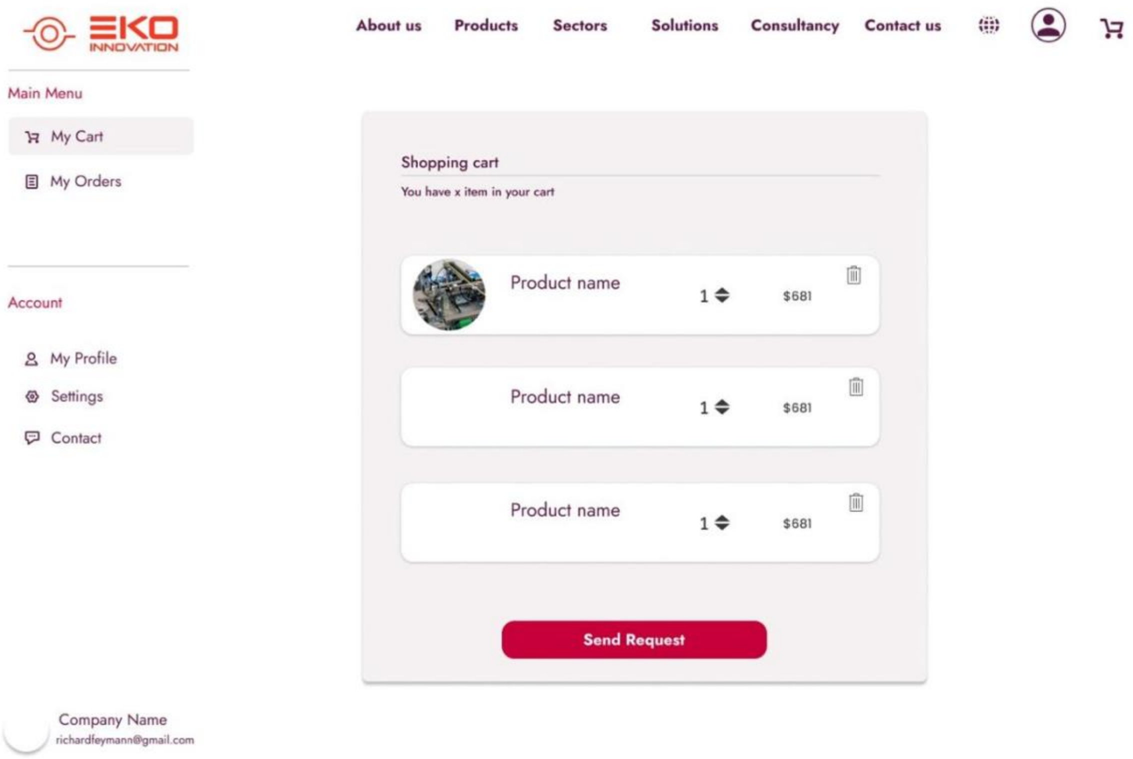
Product name	5	$540.25
Product name	5	$540.25
Product name	5	$540.25

Done

Product name	5	$540.25
Product name	5	$540.25
Product name	5	$540.25

Company Name
richardfeymann@gmail.com

Shopping Cart

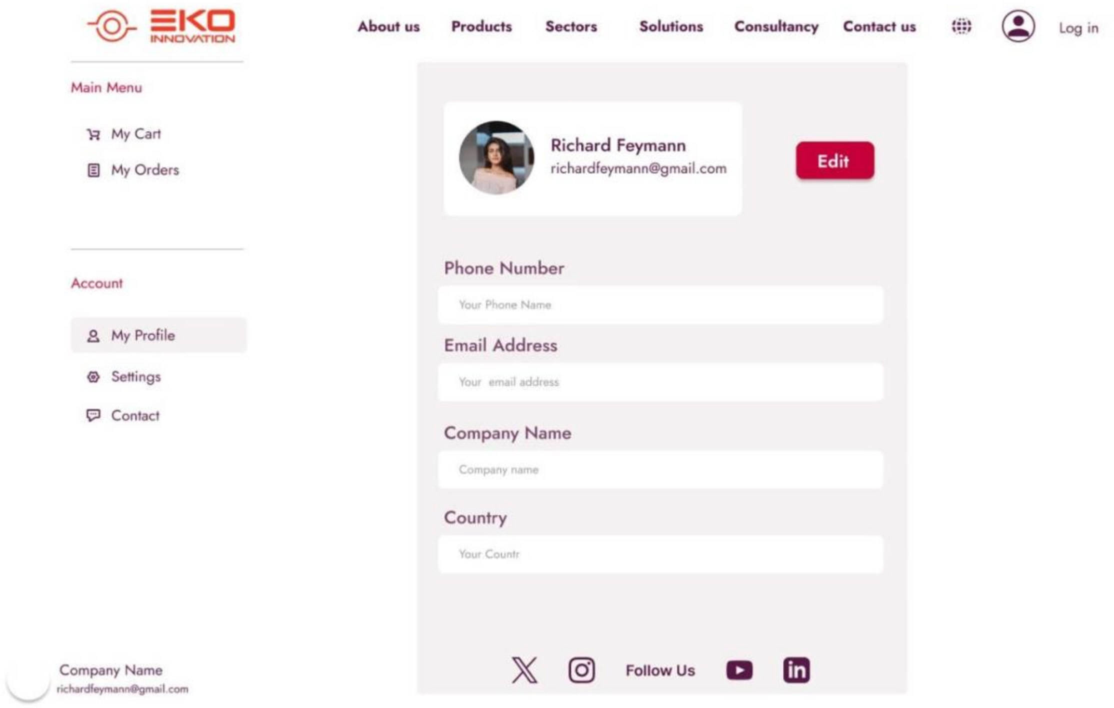


Shopping cart

You have x item in your cart

Product name	1 ⬍	$681
Product name	1 ⬍	$681
Product name	1 ⬍	$681

Profile information



Richard Feymann
richardfeymann@gmail.com

Phone Number

Your Phone Name

Email Address

Your email address

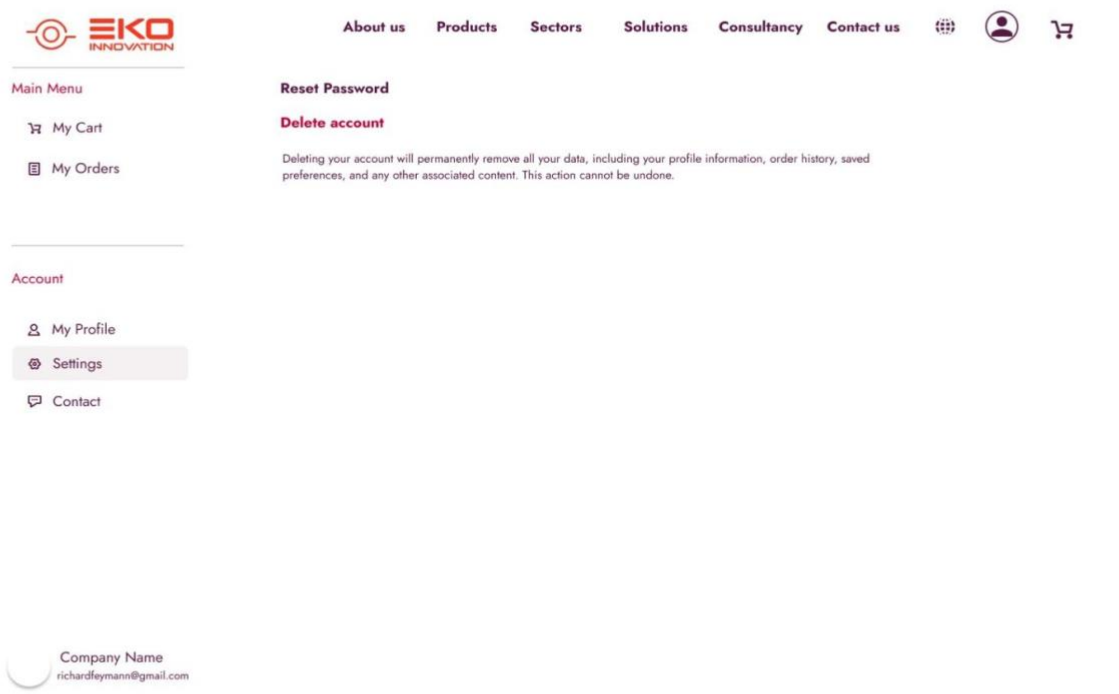
Company Name

Company name

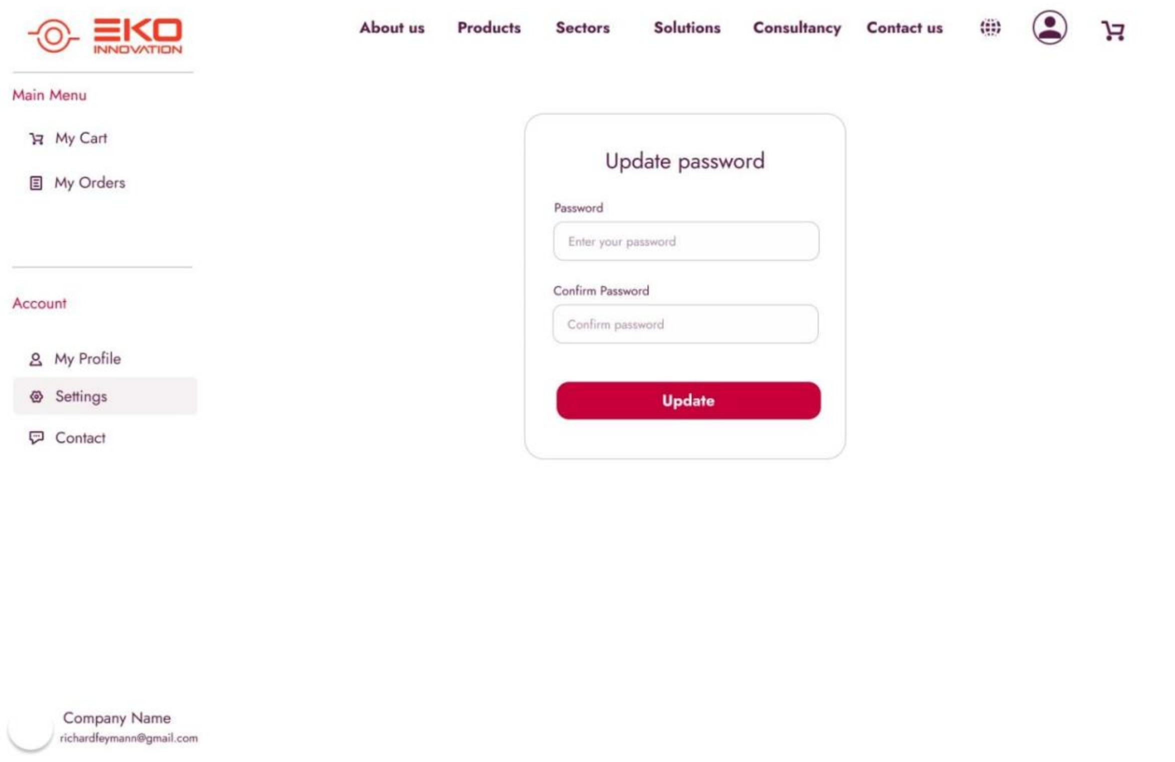
Country

Your Countr

Settings

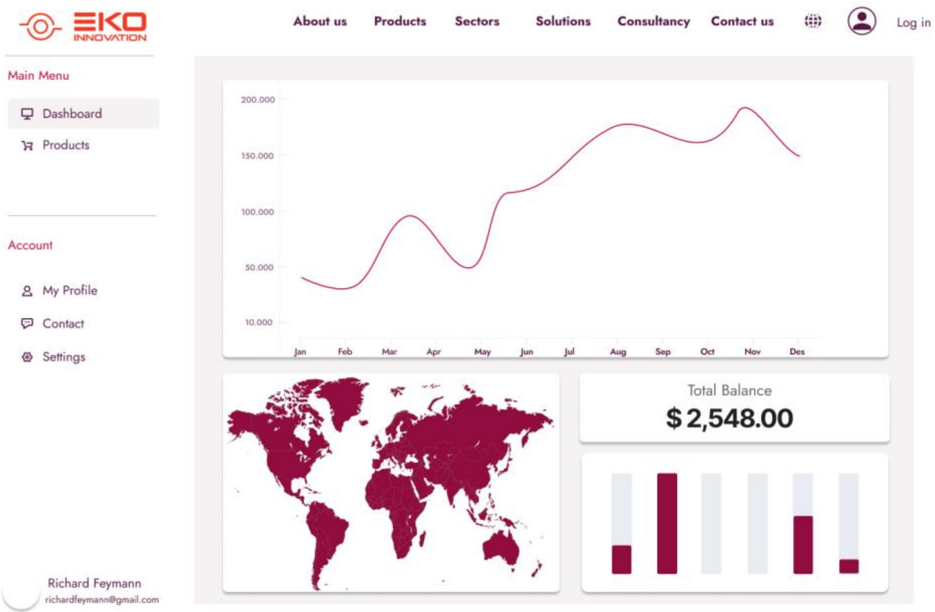


Deleting your account will permanently remove all your data, including your profile information, order history, saved preferences, and any other associated content. This action cannot be undone.

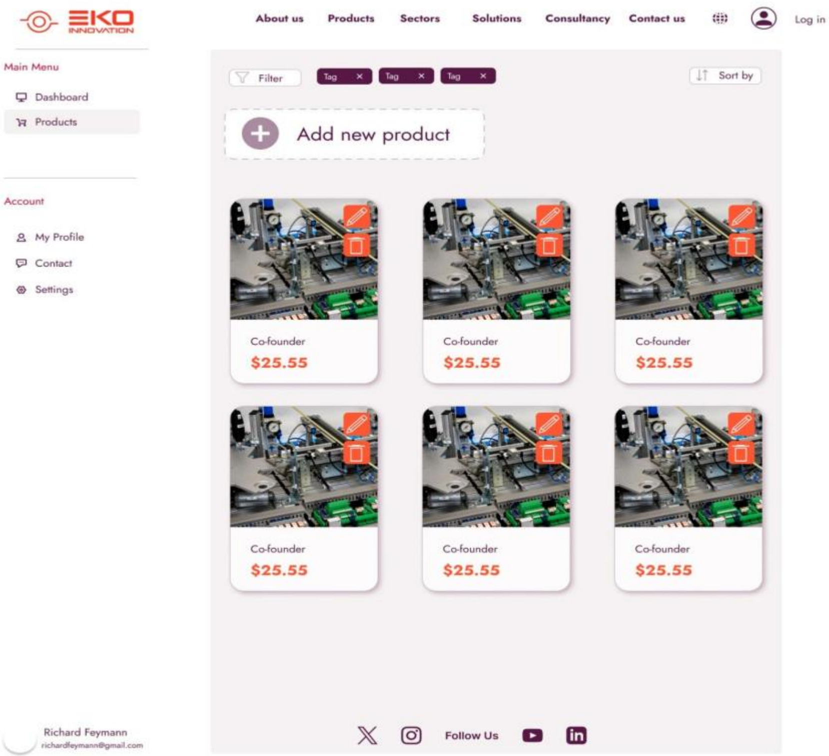


Product Admin Panel

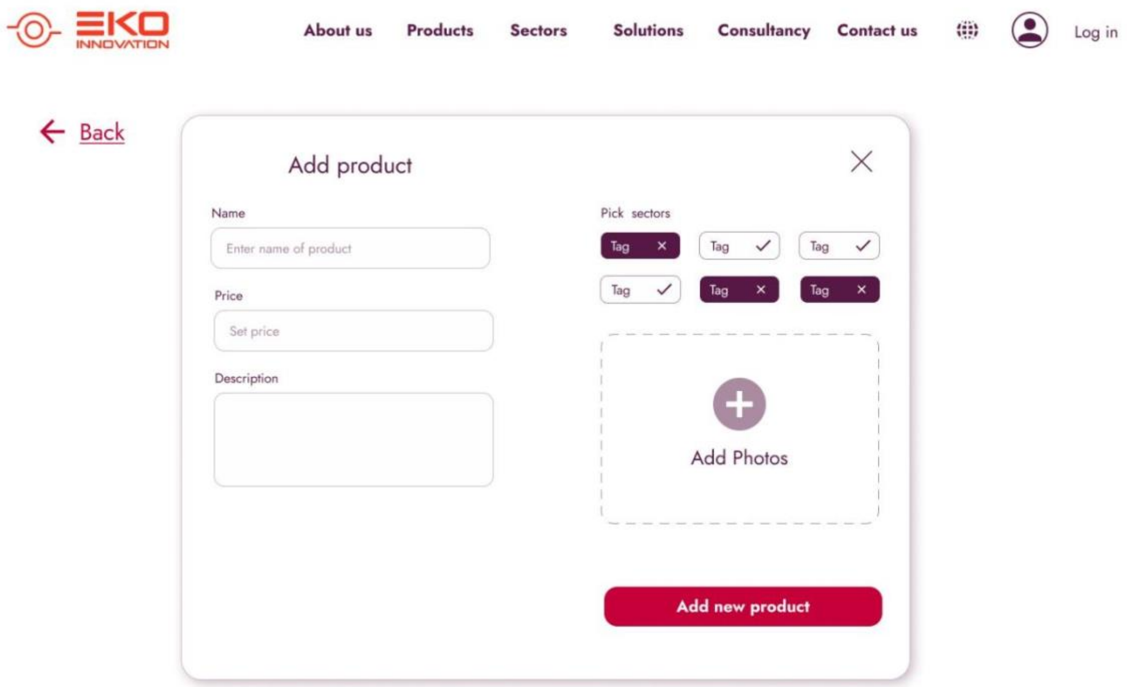
Dashboard



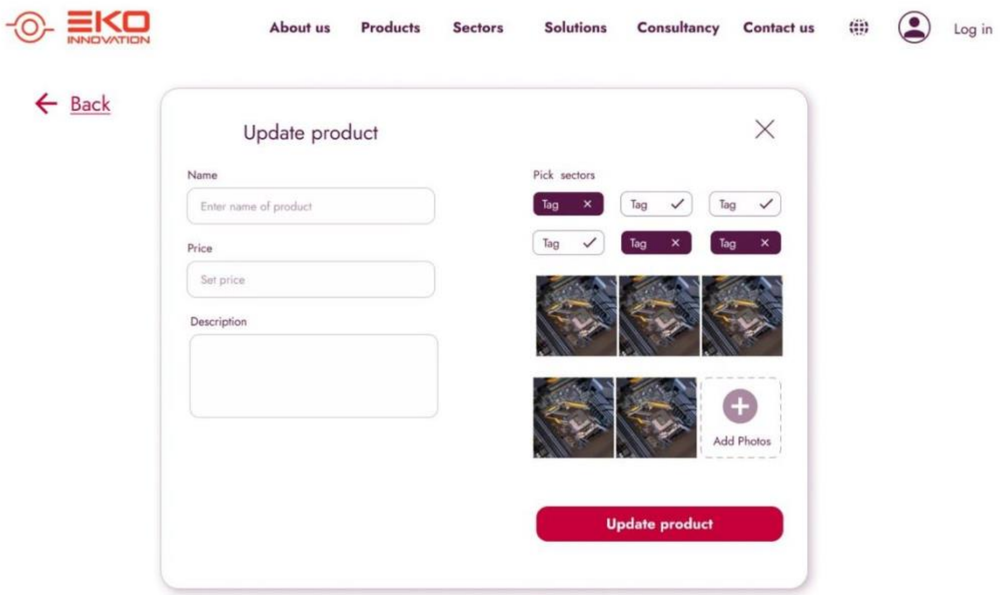
Product management



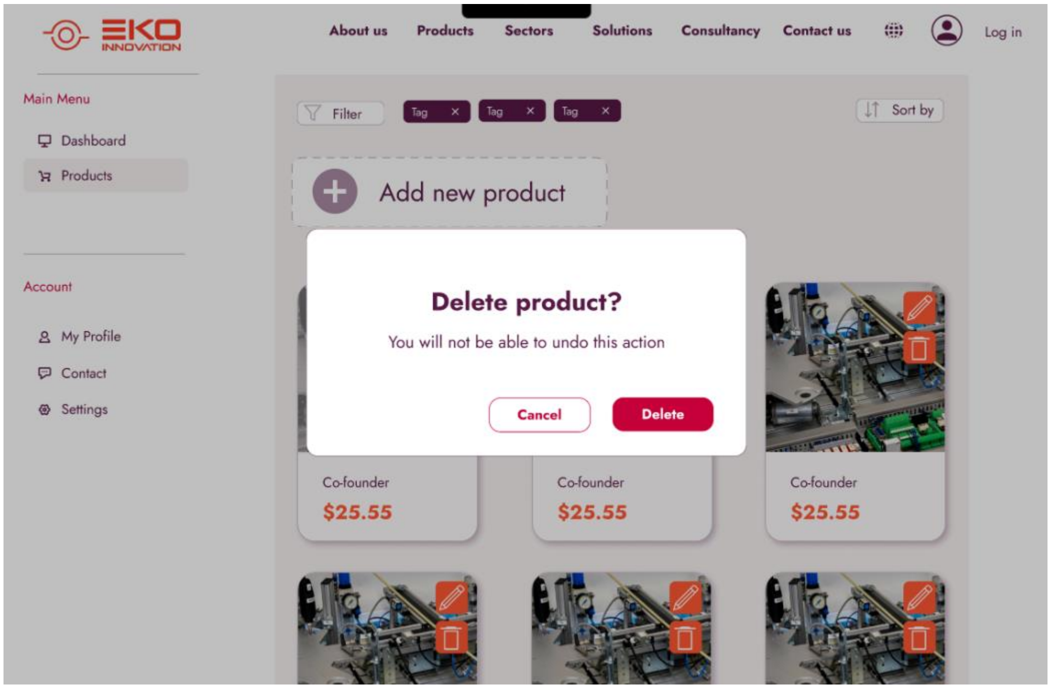
Adding new product



Update product



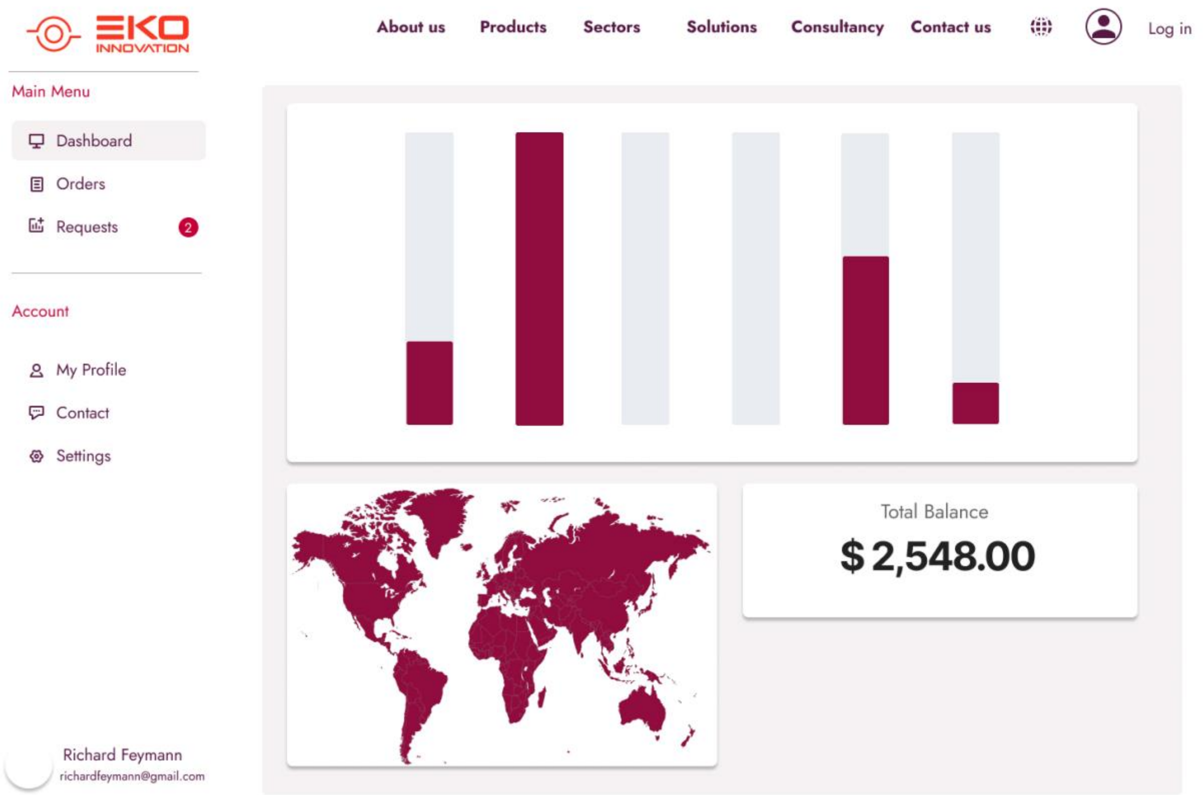
Deleting product



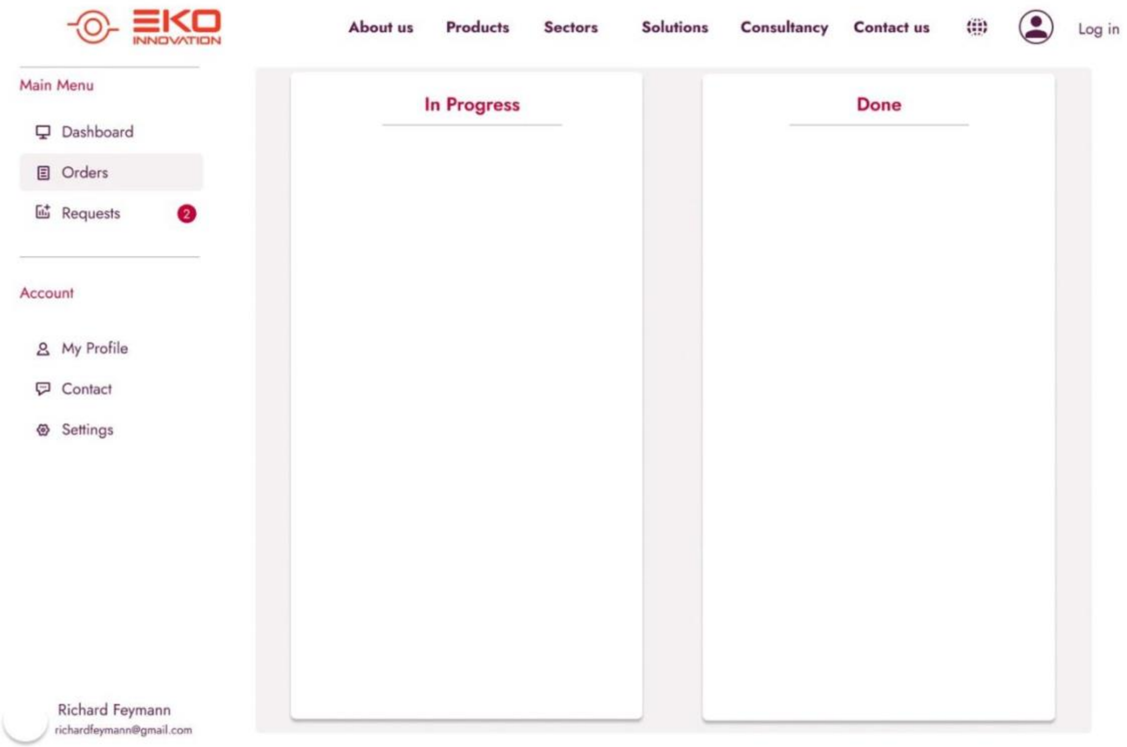
Delete product?

You will not be able to undo this action

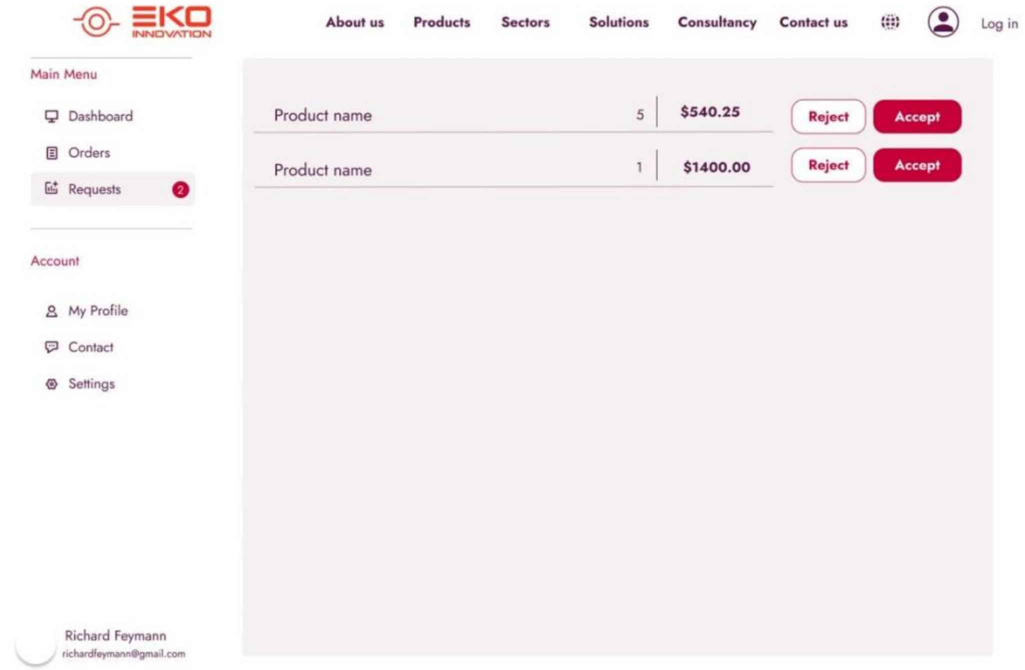
Order Admin Panel



Orders control



Request for purchase



Product name	5	$540.25
Product name	1	$1400.00

User Test Cases

Test Case 1

Test Case ID	Test 1
Test Case Description	Verify if a new user can create an account successfully
Created By	Semih Sarıkoca
Date Tested	1.11.2024
Result	Pass
Prerequisites	1. A web browser 2. URL of the website 3. User email
Test Scenario	A logged-in user successfully deletes their account.

Step #	Step Details	Expected Results	Actual Results	Pass/Fail
1	Enter valid email, password, and other required details	Account is created successfully and user receives a verification email	Account is created successfully and user receives a verification email	Pass
2	Try to create an account with an existing email	System shows an error: 'Email already registered'	System shows an error: 'Email already registered'	Pass
3	Submit registration form with missing fields	System prompts to complete required fields	System prompts to complete required fields	Pass

Test Case 2

Test Case ID	Test 2
Test Case Description	Verify if a user can delete their account successfully
Created By	Semih Sarıkoca
Date Tested	30.10.2024
Result	Pass
Prerequisites	User account with login credentials URL of the website

Test Scenario	A logged-in user successfully deletes their account.

Step #	Step Details	Expected Results	Actual Results	Pass/Fail
1	Log in to the user account	User is successfully logged in	User is successfully logged in	Pass
2	Navigate to account settings and select delete account	System prompts for confirmation of deletion	System prompts for confirmation of deletion	Pass

Test Case 3

Test Case ID	Test 3
Test Case Description	Verify if a user can log in with valid credentials
Created By	Mehmet Emir Albayrak
Date Tested	2.11.2024
Result	Pass
Prerequisites	1. A web browser 2. URL of the website 3. Existing user account
Test Scenario	A user enters valid login credentials and accesses their account.

Step #	Step Details	Expected Results	Actual Results	Pass/Fail
1	Enter valid email and password	User is logged in successfully	User is logged in successfully	Pass
2	Enter invalid password	System shows an error: 'Invalid password'	System shows an error: 'Invalid password'	Pass
3	Enter unregistered email	System shows an error: 'Invalid email address'	System shows an error: 'Invalid email address'	Pass

Test Case 4

Test Case ID	Test 4
Test Case Description	Verify if a user can log out successfully
Created By	Mehmet Emir Albayrak
Date Tested	31.10.2024
Result	Pass
Prerequisites	Logged-in user session
Test Scenario	A logged-in user logs out of their account.

Step #	Step Details	Expected Results	Actual Results	Pass/Fail
1	Click on the logout button	User is logged out successfully	User is logged out successfully	Pass

Test Case 5

Test Case ID	Test 5
Test Case Description	Verify if a user can reset their password successfully
Created By	Zerda Gül
Date Tested	31.10.2024
Result	Pass
Prerequisites	User email for password reset
Test Scenario	A user initiates password reset and successfully changes their password.

Step #	Step Details	Expected Results	Actual Results	Pass/Fail
1	Request password reset with registered email	User receives a password reset email	User receives a password reset email	Pass
2	Enter new password on the reset form	Password is successfully updated	Password is successfully updated	Pass
3	Request password reset with unregistered email	System shows an error: 'Email not registered'	System shows an error: 'Email not registered'	Pass

Test Case 6

Test Case ID	Test 6
Test Case Description	Verify if a user can update their profile information successfully
Created By	Mehmet Emir Albayrak
Date Tested	1.11.2024
Result	Pass
Prerequisites	1. Logged-in user session 2. Profile data to update
Test Scenario	A logged-in user updates their profile information, such as name and email.

Step #	Step Details	Expected Results	Actual Results	Pass/Fail
1	Navigate to profile settings and update details	Profile information is updated successfully	Profile information is updated successfully	Pass
2	Enter invalid email format	System shows an error: 'Invalid email format'	System shows an error: 'Invalid email format'	Pass

Software Requirements Spesifications

Front-End

- **HTML:** HTML provides the essential structure for web pages, ensuring that content is accessible and consistently rendered across devices. It is fundamental for creating the user interface of a web portal.
- **SCSS:** SCSS (Sassy CSS) extends traditional CSS with features like variables and nested rules, which help in maintaining a clean, modular, and consistent design—important for a scalable and visually cohesive web portal.
- **React.js:** React is a popular JavaScript library that enables the creation of interactive and dynamic user interfaces. Using React, we can build reusable components and handle data changes efficiently, creating a responsive and engaging user experience.

Back-End

- **ASP.NET:** ASP.NET is a secure and scalable web development framework ideal for handling the backend logic of a web portal. It provides strong support for object-oriented programming and offers built-in security features for user authentication and data processing, ensuring the application meets high standards of reliability and performance.

Database

- **MySQL:** MySQL is a widely used relational database that is both reliable and high-performing. It is well-suited for web portals because it can efficiently handle large volumes of data and supports complex queries, facilitating seamless integration with ASP.NET for efficient data storage and retrieval.