

CSIS 4495 - 002 - Applied Research Project

Final Report

NATIVE SPARK INTEGRATION SOLUTIONS CORP.



Valeriia Nikitina (300374609)

Contents

Introduction.....	6
The origin of the idea	6
Problem framing	6
Competitive advantage among others	7
Overview of the platform.....	8
The architecture of the system	10
The hardware and software configuration of the system	10
Hardware Requirements.....	10
Software Configuration.....	10
Changes made to the original proposal	12
Features	12
Technology.....	12
Reasons for change	13
Project Planning and Timeline	15
Planning and Proposal.....	15
System Design.....	15
Development Phase.....	15
Testing Phase.....	16
Final Submission and Presentation	16
Gantt Chart.....	17
Implemented Features.....	18
User Registration for All Three User Types.....	18
Step 1: Basic Registration Form	20
Step 2: Role-Specific Registration.....	21
Password Encryption	24

Existing Database Schema	24
Login Handling	25
Session-Based Authentication.....	26
Welcome page	27
Dynamic Navbar	28
Filter and search on Home page.....	29
Save favourites.....	29
Subscription	31
Product functionality (add, delete and update)	32
Add product	32
Update Product.....	33
Delete Product.....	34
Request custom amount	34
Python-based recommendation of products.....	37
Add to cart.....	39
Checkout Process	40
My orders summary	41
Buyers of the product.....	42
Job and Project functionality (add, delete and update)	43
Evaluation Techniques	44
Basic user	44
Business user.....	45
Reflections	46
Hours logs	47
References.....	64
Tools used	64
Downloads links:	65

Other	65
Appendix A: Installation Guide	66
1. Start XAMPP:	66
2. Open the NativeSpark Project:.....	66
3. Run the Application:	66
4. Run the Python Script:.....	66
5. Access the Application:.....	67
Appendix B: User Guide.....	68
Registration	68
Subscription	69
Home Page	70
Business user.....	71
Registration	71
My Account.....	71
Home Page	72
Cart.....	73
Services dropdown.....	74
Entrepreneur User	79
Registration	79
My Account.....	79
Home Page	80
Cart.....	82
Services dropdown.....	84
Basic User	88
Registration	88
My Account.....	88

Home Page	89
Cart.....	90
Services dropdown.....	91

Introduction

The origin of the idea

Originally developed by Valeriia Nikitina and Valeriya Saltykova, NativeSpark Integration Solutions Corp. emerged from their unique expertise and shared vision. Although the platform was initially a collaborative effort, Valeriia Nikitina will now continue to develop the idea and platform independently.

This transition marks a new chapter for NativeSpark, as Valeriia takes the lead in refining the platform's features, expanding its capabilities, and advancing its mission. By leveraging her background in data analysis and her commitment to empowering Indigenous communities, Valeriia aims to ensure that NativeSpark evolves into a robust, scalable solution that meets the current needs of its users while adapting to future challenges.

The project will maintain its original vision of fostering Indigenous entrepreneurship, with a strong focus on enhancing user experience, integrating advanced technologies, and exploring growth opportunities in the United States market. Valeriia is dedicated to preserving the collaborative spirit of the initiative while elevating the platform to new heights.

Problem framing

NativeSpark is dedicated to helping Indigenous communities overcome barriers to participation in industry. Many Indigenous entrepreneurs lack access to essential resources, mentorship, and training necessary for successful competition [1]. Additionally, Indigenous products often struggle to reach mainstream markets, and cultural and regulatory obstacles further complicate participation.

Indigenous entrepreneurs face several systemic barriers that impede their ability to engage effectively in industrial and commercial opportunities, including:

- Many Indigenous communities are situated in remote areas, making it challenging for entrepreneurs to access the funding, tools, and infrastructure needed to grow their businesses.
- Entrepreneurs frequently lack mentors who understand their unique challenges, as well as training programs tailored to their cultural and business contexts.

- Indigenous products and services often encounter difficulties navigating regulatory frameworks and facing cultural biases that prevent them from accessing mainstream markets.
- Ongoing economic exclusion has perpetuated cycles of poverty, limiting opportunities for Indigenous entrepreneurs to thrive.

These challenges are further exacerbated by the impact of social determinants of health, as highlighted in research by Richmond et al. (2017) in *Social Science & Medicine*. The study underscores that structural inequities, including limited access to education, employment, and economic opportunities, disproportionately affect Indigenous communities. These inequities diminish quality of life and hinder participation in economic systems, perpetuating cycles of poverty in marginalized communities [2, 3].

Competitive advantage among others

Existing platforms like the Indigenous Industrial and Contracting Network (IMCN) and Indigenous Business Development Services (IBDS) have made significant strides in supporting Indigenous entrepreneurs. However, they lack advanced technological features that could greatly enhance their effectiveness.

IMCN offers resources and networking opportunities but does not utilize AI-driven tools to optimize connections between businesses and entrepreneurs. On the other hand, IBDS emphasizes business development services but does not make use of predictive modeling or natural language processing (NLP) to improve decision-making and communication [4, 5].

These limitations hinder these platforms from effectively meeting the complex needs of Indigenous entrepreneurs, such as identifying market opportunities, facilitating collaboration, and offering personalized recommendations. NativeSpark aims to bridge these gaps by integrating:

- AI-Powered Matchmaking: Automatically connecting businesses with the most relevant Indigenous entrepreneurs based on their skills, location, and project requirements.
- NLP Capabilities: Enabling smooth communication across diverse linguistic and cultural backgrounds.

By addressing these gaps, NativeSpark strives to create a more comprehensive and inclusive platform for Indigenous entrepreneurs.

Overview of the platform

The platform will serve as a bridge connecting three distinct types of users—Indigenous Peoples, businesses, and regular consumers—while also integrating an administrative role for monitoring and oversight. Its core mission is to empower Indigenous communities by providing a space to showcase their craftsmanship, connect with opportunities, and facilitate transactions in an engaging and supportive environment.

Indigenous Peoples will be central to the platform, showcasing their unique skills and products through detailed profiles that include craftsmanship information and photos. They can sell items individually or in bulk and browse job postings and bulk order requests from businesses. A social networking feature will promote visibility through likes and comments.

Businesses can place large-scale orders or post projects, specifying requirements like quantity and materials, and will pay a commission on transactions. Communication with sellers for customizations is encouraged.

Regular consumers can purchase unique, handmade items, also paying a commission. They can communicate with sellers for details or customizations.

The administrator role is vital for platform integrity, overseeing user activity, moderating content, and resolving disputes to ensure a positive experience.

Social Networking for Indigenous Profiles:

Users will create portfolio-like profiles with photos, descriptions, and reviews, promoting community engagement and talent recognition.

Custom Requests:

Business users have the option to request specific quantities of products using a custom order form linked from the product details page.

Recommendation Engine:

A Python-based recommendation system suggests similar products by utilizing a cosine similarity algorithm implemented with scikit-learn.

Entrepreneur Insights:

Entrepreneurs can access information about buyers who have purchased their products, helping to foster trust and facilitate communication.

AI-Powered Matchmaking:

AI will connect businesses with Indigenous individuals based on skills, ratings, location, and availability, while consumers receive personalized product recommendations using historical data.

Integrated Chatbot with Escalation to Admins:

A chatbot will address common inquiries and guide users, escalating unresolved issues to admins for further assistance.

Admin Dashboard:

Administrators will monitor transactions, moderate content, and manage chatbot escalations to ensure a safe and effective platform.

The architecture of the system

In the development of NativeSpark's platform, we will adopt the Model-View-Controller (MVC) architecture model coupled with MySQL integration. This architectural approach is widely recognized and utilized in web application development due to its several advantages in terms of organization, scalability, and maintainability.

The MVC architecture separates an application into three interconnected components: Model, View, and Controller. Each component has specific responsibilities and interacts with the others to ensure efficient functionality and seamless user experience.

The hardware and software configuration of the system

Hardware Requirements

- Processor: Intel Core i5 or higher processors are recommended for smooth performance during development tasks.
- Memory (RAM): A minimum of 8GB RAM is recommended to handle the resource intensive tasks of running servers, databases, and development tools simultaneously.
- Network Connectivity: Stable internet connectivity is essential for accessing external resources, libraries, and version control systems during development.

Software Configuration

1) Development Environment:

- Integrated Development Environment (IDE): IntelliJ IDEA will be used as the primary IDE for Java development. It provides robust features for Java programming, Spring framework support, and seamless integration with version control systems. (Douglas College Credentials)
- Version Control: Git will be utilized for version control management, allowing collaborative development, code review, and version tracking.
- Build and Dependency Management: Maven will be used for managing dependencies and building the project.

2) Programming Languages and Frameworks:

- Java: The backend logic and business rules of the application will be developed using Java programming language.
- Spring Framework: Specifically, the Spring Boot framework will be used for rapid application development, dependency injection, and MVC architecture implementation.
- Thymeleaf: Thymeleaf will serve as the templating engine for generating dynamic HTML content in the View layer.
- Bootstrap: Bootstrap will be used for front-end styling, responsiveness, and user interface design.
- Python: Python is utilized for data seeding, database interactions, and developing a product recommendation engine.
- SQLAlchemy: This Python library is used for Object-Relational Mapping (ORM) access to MySQL database tables and models.
- Faker: Faker is a Python library that generates mock data (such as users, products, transactions, etc.) for testing and development purposes.
- scikit-learn: This is Python's machine learning library, which is employed to implement the product recommendation engine based on similarity metrics.
- JavaScript: JavaScript allows for interactive client-side features, such as save/unsaved toggle buttons and dynamic updates to the user interface.
- HTML & CSS: These technologies are used in conjunction with Thymeleaf and Bootstrap to create the markup structure and apply custom styling.

3) Database Management System (DBMS):

- MySQL: The MySQL relational database management system will be used to store and manage application data. It provides a robust and scalable solution for structured data storage.

Changes made to the original proposal

Features

Original	Final
AI-Powered Matchmaking	Not implemented
Integrated Chatbot	Not implemented
Admin Dashboard	Not implemented
Social networking features	Not implemented
Custom product requests	Added
Saved listings for users	Added
Search, filter functionality	Added
Product recommendation system	Added
Registration of 3 user types	Done
Business (add/edit/delete job/project)	Done
Indigenous Entrepreneur (add/edit/delete product)	Done
Full product lifecycle (add to cart different items, checkout)	Done

Technology

Original	Final
AI libraries: OpenAI API	Not used
Chatbot integration (Bootstrap widgets)	Not used

Python, Python-based tools	Added
Spring Boot, Thymeleaf, Bootstrap	Used
Programming languages (Java, JavaScript)	Used
MySQL	Used

Reasons for change

1. AI Matchmaking and Chatbot

Reason: The complexity of implementing these features, combined with limited time, made it challenging. They required natural language processing and third-party integrations (such as OpenAI), which would have significantly expanded the project's scope.

Justification: The focus shifted to completing the core platform functionality and product flows first before exploring advanced AI features.

2. Admin Dashboard

Reason: Priority was given to user-facing features to ensure that at least one end-user functionality was fully developed.

Justification: Given the scope and timeline of the semester project, the development of admin functionality was deferred to a later phase after project submission.

3. Addition of a Python-Based Recommendation

Reason: To provide meaningful user personalization without depending on OpenAI.

Justification: A lightweight, local solution utilizing cosine similarity and category/price metrics was implemented to offer relevant product suggestions, which was more feasible within the available timeframe.

4. New Simple Features (Search, Filters, Saved Listings)

Reason: These features were identified during development as critical for enhancing user experience.

Justification: Based on typical e-commerce platforms and feedback from peers and instructors, these features were prioritized to make the platform more functional and realistic.

5. Technology Usage Shift

Reason: Some proposed tools were deemed unnecessary or overly complex for the final project scope.

Justification: Simpler and more manageable solutions were adopted, such as writing the recommendation engine in Python instead of using external APIs.

6. Custom Product Order Requests Feature

Reason: Businesses have expressed a need to request specific quantities or variations of products directly from Indigenous sellers. Additionally, when stock runs out, basic users and other Indigenous entrepreneurs should have the option to place orders for custom quantities.

Justification: This feature enhances support for bulk orders and B2B interactions, aligning with the platform's mission to empower Indigenous entrepreneurship by fostering direct and customizable engagement between businesses.

Project Planning and Timeline

Planning and Proposal

Milestone: Proposal Approval (Jan 26)

Deliverables: Approved proposal document, initial project plan, and timeline allocation.

System Design

Milestone: Finalize System Architecture (Feb 9)

Deliverables: Database schema and Interface design.

Development Phase

Milestone: Initial Working Demo (Feb 24)

Deliverables: Partially functional prototype demonstrating core features and database integration, Video showing a demo of the implementation, midterm report.

Milestone: Advanced Feature Implementation (Mar 30)

Deliverables:

1. User Authentication and Role Management:

- Registration and login for users;
- Role-based access control: Entrepreneur, Buyer, Business;
- Dynamic navbar and page visibility based on user role.

2. Product Listing & Details:

- Products shown on the home page;
- Each product links to a detail page.

3. Saved Listings Feature:

- Save/unsave functionality for:
 - Products;
 - Jobs;
 - Projects;

- Saved items visible in a "Saved" section per user.

4. Job and Project Postings:

- Jobs and Projects posted by businesses visible to Entrepreneurs.

5. Search Functionality:

- Keyword-based search filtering.

6. Filtering:

- Products filterable by dynamically loaded categories from the database;
- Jobs filterable by employment type (Full-time, Part-time, Internship).

7. Custom Order Request Form:

- Businesses and other users can submit requests for custom product quantities.

8. Recommendation Of The Product;

9. Full Product Lifecycle: From Browsing to Purchase:

- **Browsing Products:**
- **Adding to Cart:**
- **Cart Management:**
- **Checkout Process:**
- **Order History & Details.**

Testing Phase

Milestone: Refinement, testing and design (April 6)

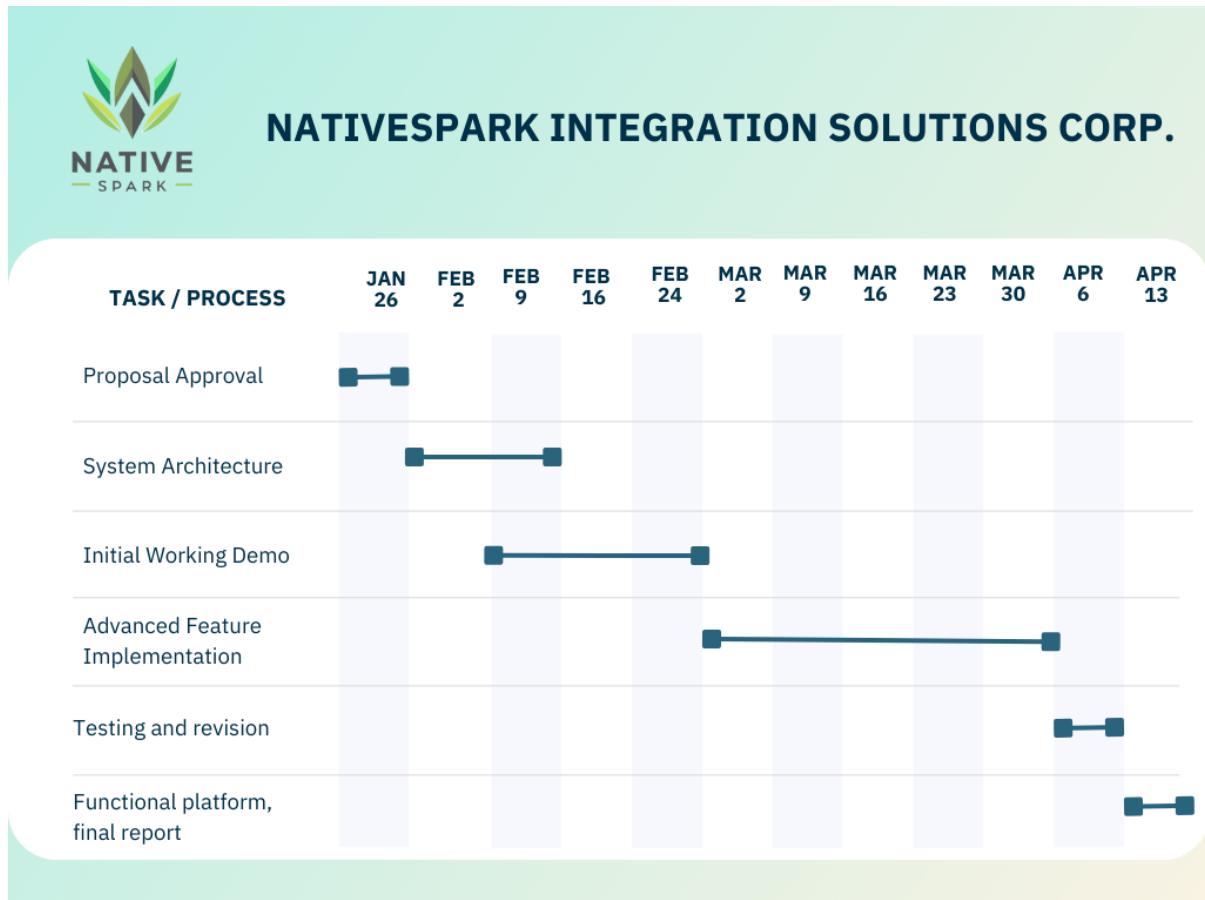
Deliverables: Usability testing and revision.

Final Submission and Presentation

Milestone: Final Submission and Presentation (Apr 13)

Deliverables: Functional platform, final report.

Gantt Chart



Implemented Features

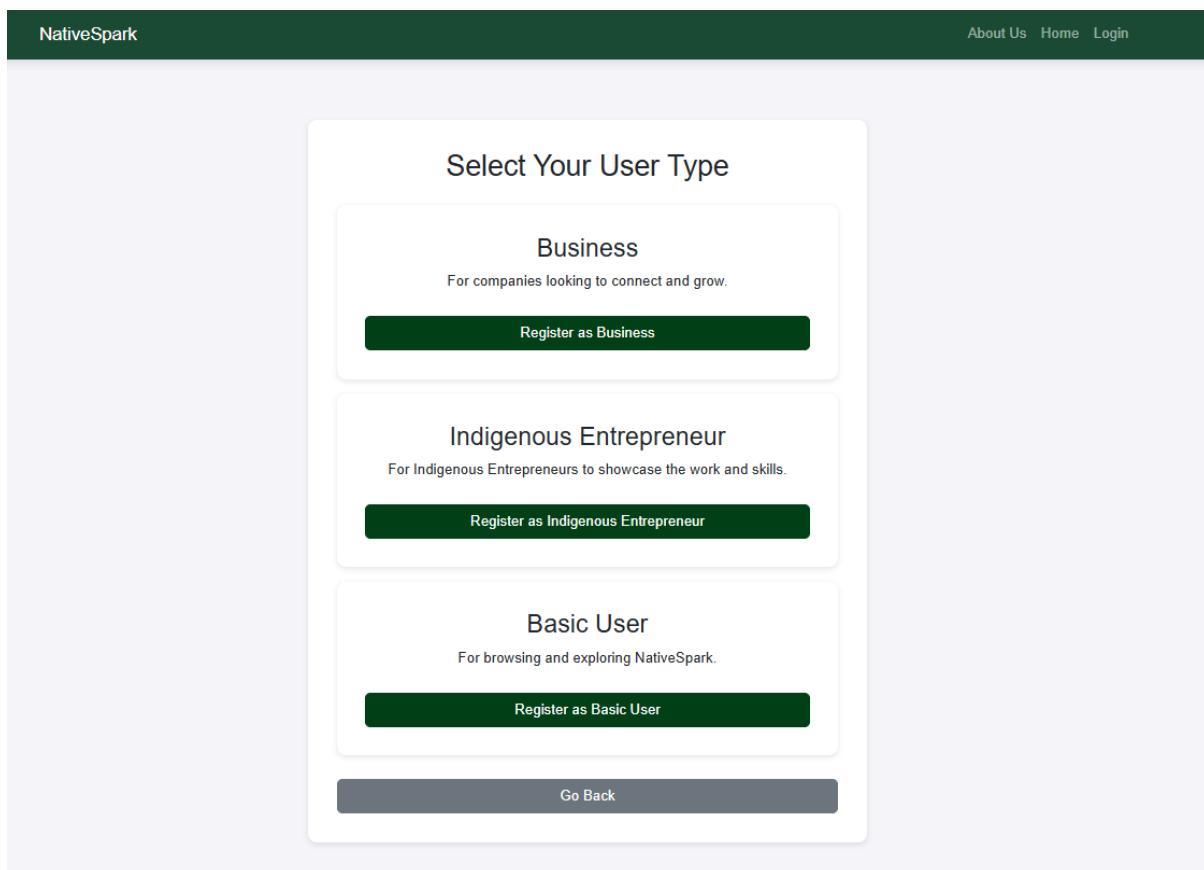
User Registration for All Three User Types

The platform supports registration for three different user roles:

- **Basic User**
- **Business User**
- **Entrepreneur User**

The registration process consists of **two steps**:

- **Step 1:** The user provides basic details such as email and password.
- **Step 2:** Additional details are collected based on the user type.

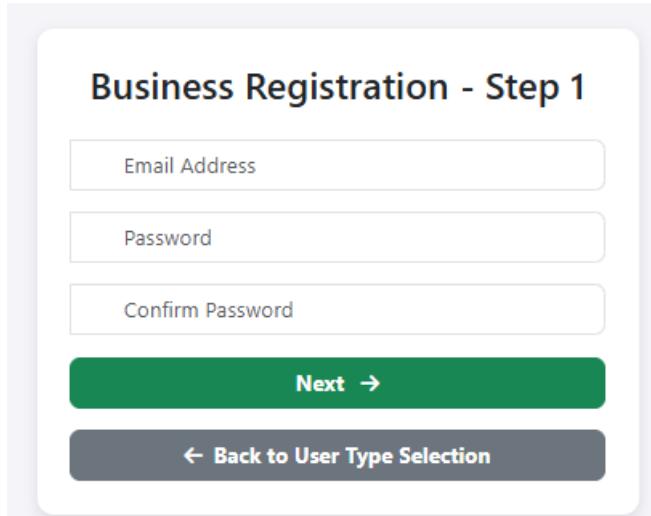


```
9     public class UserRegistrationController {  
7       @PostMapping("/register/user-type")  ↳ valerianik  
8       public String selectUserType(@RequestParam("userType") String userType, HttpSession session) {  
9           session.setAttribute("selectedUserType", userType);  
0  
1           if ("BUSINESS".equals(userType)) {  
2               return "redirect:/register/business";  
3           }  
4           else if ("ENTREPRENEUR".equals(userType)) {  
5               return "redirect:/register/entrepreneur";  
6           } else if ("BASIC".equals(userType)){  
7               return "redirect:/register/basic";  
8           }  
9  
0           return "redirect:/register/basic";  
1       }  
2  
3       @GetMapping("/register/business")  ↳ valerianik  
4       public String showBusinessRegistrationForm() {  
5           return "business-register"; //  
6       }  
7  
3       @GetMapping("/register/entrepreneur")  ↳ valerianik  
4       public String showEntrepreneurRegistrationForm() {  
5           return "entrepreneur-register"; //  
6       }  
7  
3       @GetMapping("/register/basic")  ↳ valerianik  
4       public String showBasicRegistrationForm() {  
5           return "basic-register"; //  
6       }  
7   }
```

Step 1: Basic Registration Form

Each user registers with a unified form, where they select their role and provide credentials.

Example of Business Registration:



The image shows a registration form titled "Business Registration - Step 1". It contains three input fields: "Email Address", "Password", and "Confirm Password". Below the fields is a green "Next →" button. Underneath the button is a grey "← Back to User Type Selection" button.

```
@PostMapping("/step1")  ✎ valerianik *
public String registerBusinessUserStep1(
    @RequestParam String email,
    @RequestParam String password,
    @RequestParam String confirmPassword,
    HttpSession session) {

    if (!password.equals(confirmPassword)) {
        return "redirect:/register/business?error=password_mismatch";
    }

    User user = userService.createUser(email, password, userType: "BUSINESS");
    session.setAttribute("registeredUser", user);

    return "redirect:/register/business/info";
}
```

Step 2: Role-Specific Registration

After completing Step 1, users fill in additional details.

- **Basic User** – Provides a name and profile picture.

Basic User Registration - Step 2

First Name

Last Name

Tell us about yourself

Upload Profile Photo

Choose File No file chosen

Register ✓

```
@PostMapping(value="/step2") @ResponseBody
public String registerBasicUserStep2(
    @RequestParam String firstName,
    @RequestParam String lastName,
    @RequestParam String about,
    @RequestParam("photo") MultipartFile photo,
    HttpSession session) throws IOException {

    User registeredUser = (User) session.getAttribute("registeredUser");
    if (registeredUser == null) {
        return "redirect:/register/basic";
    }

    BasicUser basicUser = basicUserService.registerBasicUser(registeredUser, firstName, lastName, about, photo);
    System.out.println("✓ Entrepreneur User Registered: " + basicUser);

    session.removeAttribute("registeredUser");
    return "redirect:/login?success=basic_registered";
}
```

- **Business User** – Adds business name, description, and logo.

Business Registration - Step 2

Business Name

Business Description

 Upload Logo

Choose File No file chosen

Register 

```

@PostMapping(value="/step2")
public String registerBusinessUserStep2(
    @RequestParam String businessName,
    @RequestParam String description,
    @RequestParam("Logo") MultipartFile logo,
    HttpSession session) throws IOException {

    User registeredUser = (User) session.getAttribute("registeredUser");

    if (registeredUser == null) {
        System.err.println("⚠ ERROR: No registered user found in session!");
        return "redirect:/register/business";
    }

    System.out.println("✅ User Found in Session: " + registeredUser.getEmail());

    BusinessUser businessUser = businessUserService.registerBusinessUser(registeredUser, businessName, description, logo);

    System.out.println("✅ Business User Registered: " + businessUser);

    session.removeAttribute("registeredUser");
    return "redirect:/login?success=business_registered";
}

```

- **Entrepreneur User** – Provides personal and professional details.

Entrepreneur Registration - Step 2

First Name

Last Name

Select Indigenous Identity

Tell us about yourself

 Upload Profile Photo

Choose File No file chosen

Register 

```
@PostMapping(value="/step2")
public String registerEntrepreneurUserStep2(
    @RequestParam String firstName,
    @RequestParam String lastName,
    @RequestParam String about,
    @RequestParam String identityType,
    @RequestParam("photo") MultipartFile photo,
    HttpSession session) throws IOException {

    User registeredUser = (User) session.getAttribute("registeredUser");
    if (registeredUser == null) {
        return "redirect:/register/entrepreneur";
    }

    EntrepreneurUser entrepreneurUser = entrepreneurUserService.registerEntrepreneurUser(registeredUser, firstName, lastName, about, identityType, photo);

    System.out.println("✓ Entrepreneur User Registered: " + entrepreneurUser);

    session.removeAttribute("registeredUser");
    return "redirect:/login?success=entrepreneur_registered";
}
```

Password Encryption

To enhance security, all passwords are hashed using BCrypt prior to being stored in the database. This method prevents the storage of plaintext passwords, thereby reducing the risk of data breaches.

```
public User createUser(String email, String password, String userType) { 3 usages • valerianik *
    if (userRepository.findByEmail(email).isPresent()) {
        throw new IllegalArgumentException("Email is already in use.");
    }

    User user = new User(email, passwordEncoder.encode(password), userType);
    return userRepository.save(user);
}
```

password

\$2a\$10\$pOqLlmvl470Aob.IZ/0jvufhv5CTAxzDZczG.RFU8kd...

\$2a\$10\$8a5QTTRffYtw5lgikl6wvOVxN06FeOmUzK5u5tKCKQ...

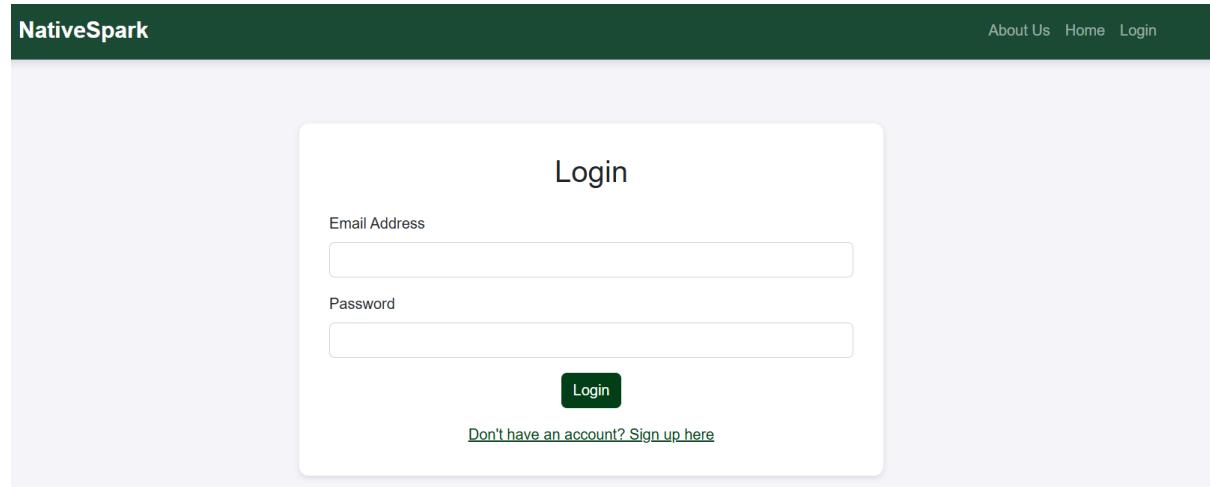
\$2a\$10\$earbZ2goQnN1SAQamLqg6u8OCIRA4D6vgfUz2LfSg7h...

Existing Database Schema

The platform follows database normalization (3NF), ensuring that user data is efficiently stored without redundancy.



Login Handling



The screenshot shows a 'Login' page with a dark header bar containing the text 'NativeSpark' on the left and 'About Us' 'Home' 'Login' on the right. The main content area has a light gray background and features a white rounded rectangle labeled 'Login'. Inside this box, there are two input fields: one for 'Email Address' and one for 'Password', both with placeholder text. Below the fields is a dark green rectangular button labeled 'Login'. At the bottom of the box, there is a link 'Don't have an account? Sign up here'.

```
@PostMapping(@RequestMapping("/login"))
public String loginUser(@RequestParam("username") String email,
                        @RequestParam("password") String password,
                        Model model) {

    System.out.println("Attempting login for email: " + email);

    Optional<User> userOptional = userService.findByEmail(email);
    if (userOptional.isEmpty()) {
        System.out.println("User not found: " + email);
        model.addAttribute("error", "Invalid email or password.");
        return "login";
    }

    User user = userOptional.get();
    if (!userService.verifyPassword(password, user.getPassword())) {
        System.out.println("Incorrect password for: " + email);
        model.addAttribute("error", "Invalid email or password.");
        return "login";
    }

    System.out.println("Authentication successful. Redirecting to account...");
    return "redirect:/account";
}
```

Session-Based Authentication

- **If a user is logged in**, they can browse restricted pages.
- **If not logged in**, they are redirected to the login page.

```
@GetMapping(@RequestMapping("account"))
public String showAccountPage(Model model) {
    org.springframework.security.core.Authentication authentication =
        SecurityContextHolder.getContext().getAuthentication();

    if (authentication == null || !authentication.isAuthenticated()) {
        System.out.println("X No authenticated user found, redirecting to login.");
        return "redirect:/login";
    }

    String email = authentication.getName();
    Optional<User> userOptional = userRepository.findByEmail(email);

    if (userOptional.isEmpty()) {
        System.out.println("X User not found in database, redirecting to login.");
        return "redirect:/login";
    }

    User user = userOptional.get();
    model.addAttribute("user", user);
    System.out.println("✓ User session loaded successfully: " + user.getEmail());

    return "account";
}
```

Welcome page

NativeSpark

About Us Home Login



Welcome to NativeSpark

NativeSpark offers specialized software for Indigenous communities in Canada and the United States.

It connects Indigenous companies and entrepreneurs with specialized resources and support.

How we see the world



NativeSpark connects Indigenous companies and entrepreneurs with specialized resources and support. Discover opportunities for economic growth and cultural preservation.

Canada provides strong government support, establishes a legal framework, demonstrates cultural diversity, and has a growing market for Indigenous goods and services. NativeSpark can leverage these strengths to work together and meet growing market demand.

Canada's global reputation as a leader in Indigenous rights and reconciliation efforts can enhance NativeSpark's credibility and visibility, attracting interest and support from international stakeholders and partners. With the support of the Canadian government, NativeSpark will create a specialized marketplace that will empower Indigenous entrepreneurs, promote economic growth, and preserve cultural heritage.

Our Mission & Vision



Our Mission

We are committed to helping Indigenous communities grow their economies by connecting them with business organizations that want to support Indigenous industrial projects.

Our Vision

Our vision is to create a web-based platform where Indigenous communities can collaborate, receive support, and access opportunities for inclusive economic development and cultural preservation.



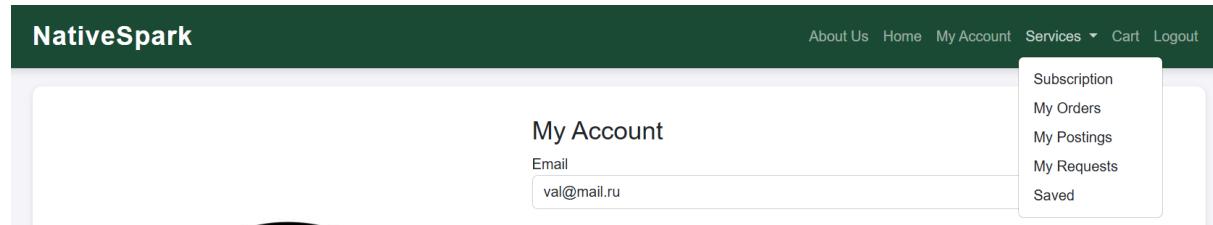
Dynamic Navbar

If the user is not logged in, the "My Account" option is hidden in the navigation bar.

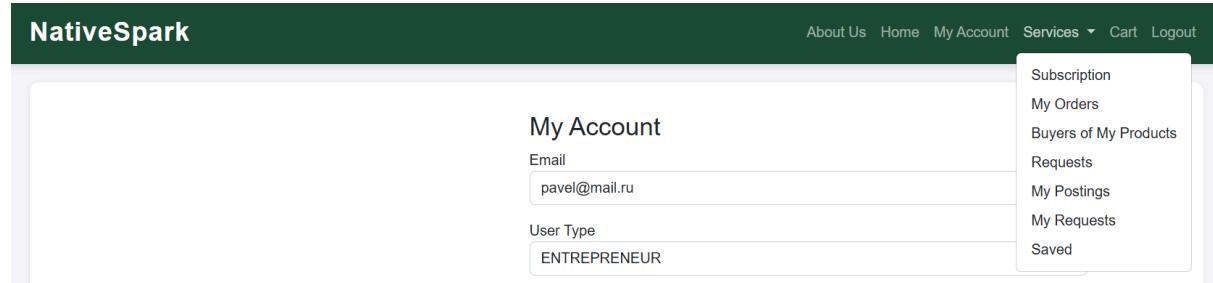


When the user is logged in more options are available:

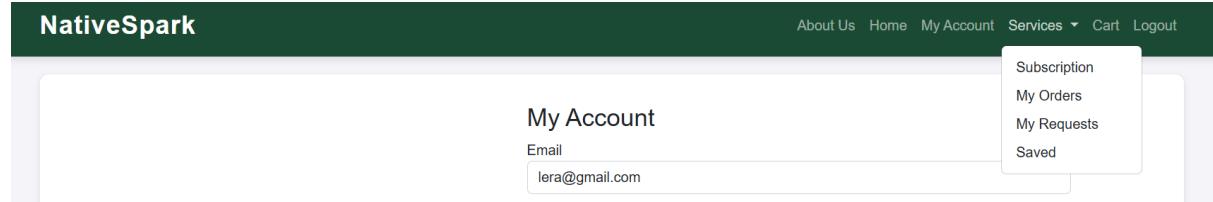
- Business



- Entrepreneur



- Basic



Filter and search on Home page

The screenshot shows the NativeSpark Marketplace homepage. At the top, there's a dark green header bar with the "NativeSpark" logo on the left and navigation links like "About Us", "Home", "My Account", "Services", "Cart", and "Logout" on the right. Below the header is a search bar with the placeholder "Search jobs, projects, or products..." and a "Search" button. The main area is titled "Marketplace". On the left, there's a sidebar with the heading "Products" and a dropdown menu titled "All Categories" which is currently expanded to show categories like Electronics, Books, Sports, Clothing, and Home. To the right of the dropdown is a "Filter" button. A tooltip-like overlay is visible over the "Electronics" category in the dropdown. At the bottom of the sidebar, there's a description placeholder: "Description: Describe president nation identify newspaper attack writer. Hand across easy radio. Eye maybe fly. Camera born task nothing.".

The user can filter products by the relevant category. Additionally, the user can search for the desired product in the search bar using keywords that will match the product name and description.

Save favourites

This screenshot shows two saved product listings. The first listing is for a product titled "Trade Exactly" in the "Electronics" category, priced at \$395.49. It includes a "Product Image" icon, a "Heart" icon for favoriting, and a description placeholder. The second listing is for a product titled "Ago Likely" in the "Books" category, priced at \$225.83. It also includes a "Product Image" icon, a "Heart" icon, and a description placeholder. Both listings have a "View Product" button at the bottom.

This screenshot shows a single saved product listing for "Trade Exactly" in the "Electronics" category. It includes a "Product Image" icon, a "Heart" icon, a "View Product" button, and a description placeholder.

```
@PostMapping("/product/{id}/toggle-save")  ✎ valerianik
@ResponseBody
public ResponseEntity<String> toggleSave(@PathVariable Long id, Authentication auth) {
    if (auth == null || !auth.isAuthenticated()) {
        return ResponseEntity.status(HttpStatus.UNAUTHORIZED).body("Not logged in");
    }

    String email = auth.getName();
    User user = userRepo.findByEmail(email).orElse( other: null);
    if (user == null) return ResponseEntity.badRequest().body("User not found");

    Product product = productRepo.findById(id).orElse( other: null);
    if (product == null) return ResponseEntity.notFound().build();

    Set<User> savedUsers = product.getSavedByUsers();
    String responseStatus;
    if (savedUsers.contains(user)) {
        savedUsers.remove(user);
        responseStatus = "unsaved";
    } else {
        savedUsers.add(user);
        responseStatus = "saved";
    }

    productRepo.save(product);
    return ResponseEntity.ok(responseStatus);
}
```

Subscription

When the user creates the account the functionality assigns him a free subscription, later the user can change it.

The screenshot shows a 'Manage Your Subscription' interface. At the top, it says 'Current Subscription: Free'. Below are three options: 'Free' (\$0 / month), 'Basic' (\$9.99 / month, highlighted in green), and 'Premium' (\$19.99 / month). The 'Payment Details' section contains fields for Card Number (1234 5678 9012 3456), Expiry Date (MM/YY), and CVV (123). At the bottom are 'Save Subscription' and 'Back to Account Page' buttons.

```
// Determine cost based on subscription type
double cost = 0.0;
if ("Basic".equalsIgnoreCase(subscriptionType)) {
    cost = 9.99;
} else if ("Premium".equalsIgnoreCase(subscriptionType)) {
    cost = 19.99;
}

if (subscriptionOptional.isPresent()) {
    Subscription subscription = subscriptionOptional.get();
    logger.info("Updating subscription for user: " + user.getEmail() + " to " + subscriptionType);
    subscription.setSubscriptionType(subscriptionType);
    subscriptionRepository.save(subscription);
} else {
    logger.info("Creating new subscription for user: " + user.getEmail());
    Subscription newSubscription = new Subscription(user, subscriptionType);
    subscriptionRepository.save(newSubscription);
}
```

Product functionality (add, delete and update)

Add product

Create Product Listing

Product Name

Product Description

Price

QTY

Category

 Clothing

Product Image

Choose File No file chosen

Save Product Cancel

```
@PostMapping("/save")  ↳ valerianik
public String saveProductPosting(@ModelAttribute Product product,
                                  @RequestParam("imageFile") MultipartFile imageFile,
                                  Principal principal) throws IOException {
    User user = getAuthenticatedUser(principal);
    if (user == null) return "redirect:/login";

    Optional<EntrepreneurUser> entrepreneurUser = entrepreneurUserRepository.findByUser(user);
    if (entrepreneurUser.isPresent()) {
        product.setEntrepreneur(entrepreneurUser.get());
        product.setPostedDate(LocalDateTime.now());

        // ✅ Handle Image Upload
        if (!imageFile.isEmpty()) {
            File directory = new File(UPLOAD_DIR);
            if (!directory.exists()) {
                boolean created = directory.mkdirs();
                if (!created) {
                    throw new IOException("Could not create upload directory: " + UPLOAD_DIR);
                }
            }
        }
    }
}
```

```

    // ✅ Generate unique filename
    String fileName = System.currentTimeMillis() + "_" + imageFile.getOriginalFilename();
    Path filePath = Paths.get(UPLOAD_DIR, fileName);

    Files.copy(imageFile.getInputStream(), filePath);

    // ✅ Save relative image path for Thymeleaf
    product.setImagePath("/" + UPLOAD_DIR + fileName);
}

productRepository.save(product);
}

return "redirect:/my_postings"; // ✅ Redirect back to My Postings page
}

```

Update Product

The screenshot shows a web application interface for updating a product posting. The top navigation bar includes links for About Us, Home, My Account, Services, Cart, and Logout. The main content area is titled "Edit Product Posting". The form fields are as follows:

- Product Name:** Paint
- Category:** Art
- Price (\$):** 50.0
- QTY:** 5
- Product Description:** Cover the surface of (something) with paint, as decoration or protection.
- Product Image:** A file input field showing "No file chosen". Below it, a note says "Leave blank to keep the current image."
- Buttons:** "Update Product" (green button) and "Cancel"

```

@PostMapping("/update/{id}")
public String updateProductPosting(@PathVariable Long id, @ModelAttribute Product product,
                                    @RequestParam(value = "imageFile", required = false) MultipartFile imageFile) throws IOException {
    Optional<Product> existingProduct = productRepository.findById(id);

    if (existingProduct.isPresent()) {
        Product updatedProduct = existingProduct.get();
        updatedProduct.setProductName(product.getProductName());
        updatedProduct.setCategory(product.getCategory());
        updatedProduct.setPrice(product.getPrice());
        updatedProduct.setProductDescription(product.getProductDescription());
        updatedProduct.setQuantity(product.getQuantity());

        // Handle Image Upload
        if (imageFile != null && !imageFile.isEmpty()) {
            String fileName = System.currentTimeMillis() + "_" + imageFile.getOriginalFilename();
            Path filePath = Paths.get( first: "uploads/products/", fileName);
            Files.copy(imageFile.getInputStream(), filePath);
            updatedProduct.setImagePath("/uploads/products/" + fileName);
        }
    }

    productRepository.save(updatedProduct);
}

return "redirect:/my_postings";
}

```

Delete Product

```

@PostMapping("/delete/{id}")
public String deleteProductPosting(@PathVariable Long id) {
    productRepository.deleteById(id);
    return "redirect:/my_postings";
}

```

Request custom amount

The screenshot shows a product page for a LEGO item on the NativeSpark website. The product image is the classic LEGO logo. The product details include:

- Category:** Other
- Price:** \$150.0
- Available Quantity:** 10
- Posted Date:** 2025-04-11 13:06
- Description:** LEGOs are plastic building-block toys that rose to massive popularity in the mid-20th century.
- Seller:** Pavel Sitnik

Below the details, there is a quantity input field set to 1, an "Add to Cart" button, and two other buttons: "Request Custom Amount" and "Back to Home".

Request Your Custom Amount



Lego

Category: Other**Price:** \$150.0**Available Quantity:** 10

Description: LEGOs are plastic building-block toys that rose to massive popularity in the mid-20th century.

Enter Desired Amount:

Submit Request**Back to Home**

```

@PostMapping("/{id}")
public String submitCustomRequest(@PathVariable Long id,
                                  @ModelAttribute("request") CustomProductRequest request,
                                  Authentication auth) {
    Product product = productRepository.findById(id).orElseThrow();
    User user = userRepository.findByEmail(auth.getName()).orElseThrow();

    request.setProduct(product);
    request.setUser(user);
    customProductRequestRepository.save(request);

    return "redirect:/request-custom/" + id + "?success=true";
}
  
```

Custom requests are shown in account of the user.

My Custom Requests

Lego

Requested Amount: 1000**Seller:** Pavel Sitnik (pavel@mail.ru)**Date of Request:** 2025-04-11 13:19

```

@GetMapping("/{my_requests}")
public String viewMyCustomRequests(Model model, Authentication authentication) {
    User user = userRepository.findByEmail(authentication.getName()).orElseThrow();
    List<CustomProductRequest> customRequests = customProductRequestRepository.findByUser(user);
    model.addAttribute("customRequests", customRequests);
    return "my_requests";
}

```

The user will also see the request to make it

The screenshot shows a web application interface. At the top, there is a dark header bar with the text "NativeSpark" on the left and navigation links "About Us", "Home", "My Account", "Services", "Cart", and "Logout" on the right. Below the header, the main content area has a light gray background. The title "Custom Requests for My Products" is centered at the top of this area. Below the title, there is a list of requests. The first request in the list is for a product named "Lego". The details for this request are as follows:

- Requested Amount:** 1000
- Requester:** val@mail.ru
- Requested At:** 2025-04-11 13:19

```

@.GetMapping("/{requests}")
@PreAuthorize("hasAuthority('ROLE_ENTREPRENEUR')")
public String viewRequestsForMyProducts(Model model, Authentication authentication) {
    User entrepreneurUser = userRepository.findByEmail(authentication.getName()).orElseThrow();
    EntrepreneurUser entrepreneur = entrepreneurUserRepository.findByUser(entrepreneurUser).orElseThrow();

    List<CustomProductRequest> allRequests = customProductRequestRepository.findAllByProduct_Entrepreneur(entrepreneur);
    model.addAttribute("requests", allRequests);

    return "requests"; // ✅ This must match the name of your Thymeleaf template (requests.html)
}

```

Python-based recommendation of products

The program shows the recommendations of the product based on the category and the price for the product.

SQLAlchemy: This Python library is used for Object-Relational Mapping (ORM) access to MySQL database tables and models.

scikit-learn: This is Python's machine learning library, which is employed to implement the product recommendation engine based on similarity metrics.

The screenshot displays a web application interface. At the top, a dark green header bar contains the logo 'NativeSpark' on the left and navigation links 'About Us', 'Home', 'My Account', 'Services', 'Cart', and 'Logout' on the right. Below the header, the main content area features a large product detail card for a 'Lego' item. The card includes a large image of the LEGO logo, the product name 'Lego', its category 'Other', price '\$150.0', available quantity '10', posted date '2025-04-11 13:06', and a detailed description stating 'LEGOs are plastic building-block toys that rose to massive popularity in the mid-20th century.' It also shows the seller 'Pavel Sitnik' and a quantity input field with '1' and an 'Add to Cart' button. A 'Back to Home' button is at the bottom. Below this card, a section titled 'Recommendations' is shown, featuring a grid of smaller product cards. One card for 'Lego' is identical to the one above. Other recommended products include 'Air Game Sports', 'Activity Growth Home', 'Major Lay Sports', 'Issue Bill Electronics', and another 'Lego' card. Each recommended product card includes a 'View Product' button.

Script for recommendations

```
import pandas as pd
import numpy as np
from dbalchemy import (
    ProductRecommendation,
    session,
    products_table,
    mysqlengine,
    reccomended_products_table,
```

```

)
from sklearn.metrics.pairwise import cosine_similarity
from sklearn.preprocessing import MinMaxScaler

# Load products
def load_products_df():
    with mysqlengine.connect() as connection:
        query = products_table.select()
        df = pd.read_sql(query, connection)
    return df

def get_recommendations_for_product(products_df, similarity_matrix, product_id, n=2):
    product_idx = products_df.index[products_df["id"] == product_id].tolist()[0]

    similarity_scores = similarity_matrix[product_idx]
    similar_indices = similarity_scores.argsort()[:-1][1:n+1]

    recommendations = products_df.iloc[similar_indices][["id", "category", "price"]]
    return recommendations

def create_reccomendations(products_df):
    category_encoded = pd.get_dummies(products_df["category"])

    scaler = MinMaxScaler()
    price_normalized = scaler.fit_transform(products_df[["price"]])

    feature_matrix = np.hstack((category_encoded.values, price_normalized))
    similarity_matrix = cosine_similarity(feature_matrix)

    insert_data = []
    for product_idx, product_id in enumerate(products_df["id"]):
        similarity_scores = similarity_matrix[product_idx]
        similar_indices = similarity_scores.argsort()[:-1][1:6]
        for idx in similar_indices:
            insert_data.append(
                {
                    "product_id": product_id,
                    "recommended_product_id": int(products_df.iloc[idx]["id"]),
                }
            )
    with mysqlengine.connect() as connection:
        connection.execute(recommended_products_table.insert(), insert_data)

```

```

connection.commit()

def clear_reccomendations():
    session.query(ProductRecommendation).delete()
    session.commit()

def main():
    clear_reccomendations()
    products_df = load_products_df()
    create_reccomendations(products_df)

if __name__ == "__main__":
    main()

```

Add to cart

The user can add the product to cart and update the qty or delete the product from cart.

Lego

Category: Other

Price: \$150.0

Available Quantity: 10

Posted Date: 2025-04-11 13:06

Description: LEGOs are plastic building-block toys that rose to massive popularity in the mid-20th century.

Seller: Pavel Sitnik

1

[Back to Home](#)

Product Name	Description	Price	Quantity	Total	Actions
Lego	LEGOs are plastic building-block toys that rose to massive popularity in the mid-20th century.	150.0	1 <input type="button" value="Update"/>	150.0	<input type="button" value="Remove"/>
Paint	Cover the surface of (something) with paint, as decoration or protection.	50.0	1 <input type="button" value="Update"/>	50.0	<input type="button" value="Remove"/>

[Proceed to Checkout](#)

```

@PostMapping("/cart/updateQuantity")  ↳ valerianik
public String updateQuantity(@RequestParam Long cartItemId, @RequestParam int quantity) {
    CartItem cartItem = cartItemRepository.findById(cartItemId)
        .orElseThrow(() -> new RuntimeException("CartItem not found"));
    cartItem.setQuantity(quantity);
    cartItemRepository.save(cartItem);
    return "redirect:/cart";
}

@PostMapping("/cart/removeItem")  ↳ valerianik
public String removeItem(@RequestParam Long cartItemId) {
    CartItem cartItem = cartItemRepository.findById(cartItemId)
        .orElseThrow(() -> new RuntimeException("CartItem not found"));
    cartItemRepository.delete(cartItem);
    return "redirect:/cart";
}

```

Checkout Process

The screenshot shows a user interface for a checkout process. At the top, there is a dark header bar with the text "NativeSpark". To the right of the header, there are several navigation links: "About Us", "Home", "My Account", "Services ▾", "Cart", and "Logout".

The main content area is divided into three columns:

- Seller Info**: A box containing the seller's name and email.
- Product Info**: A box listing the items in the cart, including their names, prices, quantities, and totals.
- Summary**: A box displaying the total number of items and price, followed by a "Proceed to Checkout" button.

Below these columns, there is another row of boxes, likely representing additional items in the cart.

Name	Price	Quantity	Total
Lego	\$150.0	1	\$150.00
Paint	\$50.0	1	\$50.00

Shipping Information

Country:

City:

Province:

Street Address:

Postal Code:

Phone Number:

Payment Details

Card Number:

Expiration Date:

 MM/YY

CVV:

Confirm Purchase

My orders summary

My Orders

Lego

Quantity: 1

Total Price: \$150.00

Seller: pavel@mail.ru

Phone Number: 6045006272

Status: SUCCESS

Shipping Address: 216 1tbleu, van, bc, canada, 5tghb7

Date: 2025-04-11 13:34

Paint

Quantity: 1

Total Price: \$50.00

Seller: pavel@mail.ru

Phone Number: 6045006272

Status: SUCCESS

Shipping Address: 216 1tbleu, van, bc, canada, 5tghb7

Date: 2025-04-11 13:34

```

@GetMapping(@RequestMapping("/my_orders")) ✎ valerianik
public String viewOrders(Model model, Authentication authentication) {
    String email = authentication.getName();
    Optional<User> userOptional = userRepository.findByEmail(email);

    if (userOptional.isPresent()) {
        User user = userOptional.get();
        List<Transaction> orders = transactionRepository.findByBuyer(user);
        model.addAttribute(attributeName: "orders", orders);
    }

    return "my_orders";
}

```

Buyers of the product

The screenshot shows a web application interface. At the top, there is a dark header bar with the text "NativeSpark" on the left and navigation links "About Us", "Home", "My Account", "Services", "Cart", and "Logout" on the right.

The main content area has a light gray background. In the center, the heading "People Who Bought My Products" is displayed. Below the heading, there is a detailed list of a single purchase:

- Lego**
- Quantity:** 1
- Price:** \$150.0
- Paint**
- Quantity:** 1
- Price:** \$50.0
- Total Price:** \$200.0
- Buyer Email:** lera@gmail.com
- Phone:** 6045006272
- Shipping Address:** 216 ltbleu, van, bc, canada, 5tghb7
- Date:** 2025-04-11 13:34

```

@GetMapping(@RequestMapping("/buyers")) ✎ valerianik
public String viewBuyersOfMyProducts(Model model, Authentication authentication) {
    String email = authentication.getName();
    User user = userRepository.findByEmail(email).orElse(other: null);

    if (user == null) {
        return "redirect:/login";
    }

    List<Transaction> sales = transactionRepository.findBySeller(user); // ✅ correct lookup
    model.addAttribute(attributeName: "sales", sales);

    return "buyers";
}

```

Job and Project functionality (add, delete and update)

The screenshot shows a web application interface for 'NativeSpark'. At the top, there's a dark green header bar with the 'NativeSpark' logo on the left and navigation links for 'About Us', 'Home', 'My Account', 'Services', 'Cart', and 'Logout' on the right. Below the header, the main content area has a light gray background. It features a title 'My Postings' at the top center, with a 'Back to Account Page' button to its right. Underneath, there are two tabs: 'Job Postings' (which is highlighted in blue) and 'Project Postings'. A 'Job Postings' section is displayed, containing a single listing for a 'Data Analyst' position. The listing includes details like company ('Anyvisa services corp.'), location ('Vancouver'), employment type ('Full-time'), salary ('\$100000.0'), required experience ('4'), required skills ('Structured Query Language (SQL)', 'Microsoft Excel', 'Critical thinking', 'R or Python statistical programming', 'Data visualization', 'Presentation skills', 'Machine learning'), posted on ('2025-04-11 13:43'), and a description of the role. To the right of the listing are 'Edit' and 'Delete' buttons. In the top right corner of the main content area, there's a green 'Add Job Posting' button.

This screenshot shows the same 'NativeSpark' website interface, specifically the 'Project Postings' section. The layout is identical to the 'Job Postings' page above it. The main content area has a light gray background with a title 'My Postings' and a 'Back to Account Page' button. Below these are the 'Job Postings' and 'Project Postings' tabs, with 'Project Postings' currently selected. A 'Project Postings' section is displayed, containing a single listing for 'Business Plan Development'. The listing includes details like client ('Anyvisa services corp.'), budget ('\$7000.0'), deadline ('2025-04-30'), required skills ('Defining your business idea', 'Clarifying the market and competitive landscape'), scope ('A business plan should ideally be reviewed and updated periodically to reflect achieved goals or changes in direction. An established business moving in a new direction might even create an entirely new plan.'), posted on ('2025-04-11 13:47'), and a description of the role. To the right of the listing are 'Edit' and 'Delete' buttons. In the top right corner of the main content area, there's a green 'Add Project Posting' button.

Evaluation Techniques

The mixed-method evaluation was made to identify strengths in the current implementation and uncover areas for future enhancement, ensuring that the platform aligns closely with the expectations and needs of its target audience.

Basic user

Also, besides the survey the discussion was made afterwards, I was asked to help with the development of other platform (due to my knowledge of how to make the platform work).

1. How easy was it to navigate the platform and find what you were looking for?
2. Did the product listings (images, descriptions, pricing) provide enough information for you to make a decision?
3. How likely are you to purchase Indigenous-made products through NativeSpark?
4. What features do you find most valuable as a basic user?
5. If a product is out of stock, how useful is the 'Request Custom Amount' feature for you?
6. What additional features would make you more likely to use this platform regularly?
7. How comfortable were you with the design and layout of the site?
8. Did you experience any issues while browsing, searching, or saving items?
9. Based on what you've seen and the idea presented, how likely are you to recommend NativeSpark to others?

Most users reported that the platform is easy to navigate and visually appealing, demonstrating success in creating an intuitive user interface. Key features such as product search, filters, the "Save" (heart) button, and the "Request Custom Amount" function were particularly highlighted as useful, contributing to a positive user experience. Additionally, the personalized recommendation engine was appreciated for helping users discover relevant products.

However, aside from satisfaction with current functionalities, users expressed a strong desire for more advanced tools to enhance decision-making and shopping confidence. Suggestions included implementing a user-driven review and rating system to evaluate products and sellers based on community feedback, as well as expanding the filtering system to allow for more precise sorting by item characteristics (e.g., size, color, price range, location). This feedback

reflects a growing expectation for customization and trust-building features, which are becoming standard in modern e-commerce platforms.

Business user

1. Is there anything missing from NativeSpark that would make your work easier?
2. If you could add one feature to the platform, what would it be?
3. What do you wish was different or better about the current interface or process?
4. How easy is it to post your jobs or projects on the platform?
5. Is the "Save" feature (heart icon) useful for you? Why or why not?
6. How do you feel about the "My Orders" page - is it giving you enough detail?

Overall, user appreciated the platform's functionality and user interface, finding it generally intuitive and effective for posting jobs or projects. While the onboarding experience for returning users was considered smooth, there were suggestions for enhancing the onboarding process for first-time users. Suggestions included adding templates or guided prompts during job or project creation.

A consistent theme in the feedback was the need for improved communication and management tools. Business users found the "Save" (heart) feature useful for tracking profiles, but expressed interest in expanding its capabilities with features such as tagging or folder creation. Additionally, they recommended integrating a live chat option to facilitate real-time communication with Indigenous creators or support staff, which would enhance responsiveness and reduce friction in transactions.

Feedback regarding the "My Orders" page indicated a desire for more detailed information and tracking. Business users requested clearer timelines, a visual progress tracker, and the ability to leave private notes on each transaction to enhance internal project management. Another important feature requested was a way to view and manage candidates who expressed interest in job or project postings, which would streamline hiring and collaboration efforts.

In conclusion, while business user was satisfied with the platform's current offerings, their suggestions highlight the need for more robust operational tools and process automation. Prioritizing these enhancements in future development phases will not only improve the efficiency of business users but also contribute to a more dynamic and scalable marketplace environment on NativeSpark.

Reflections

Throughout the development of NativeSpark, I faced several technical and conceptual challenges that greatly contributed to my learning. One of the major obstacles was related to setting up the backend environment. Initially, I planned to use XAMPP for MySQL database management, but it stopped responding midway through the development process. This issue led me to switch to a standalone MySQL installation, which turned out to be more stable and reliable for integrating with my Spring Boot application.

An important insight I gained during the project came from implementing the recommendation engine. Before this experience, I didn't realize that Python could be used in a modular, object-oriented way, similar to Java. I had previously thought that Python was only suitable for scripting or simple automation. However, as I worked on generating data using the Faker library, building a recommendation engine with scikit-learn, and managing the database with SQLAlchemy, I discovered a strong interest in using Python to work with datasets and derive insights from them—a field I would love to continue exploring.

The most satisfying part of the project was seeing the entire platform come together—from the frontend interface to database interactions and custom product requests. It was rewarding to witness a real, functional system where different users could interact seamlessly. Looking ahead, I'm excited about the idea of deploying the platform to a public domain. I would love to learn how to deploy it on cloud platforms like AWS or Heroku so that real users can access NativeSpark through a live domain and engage with it in a real-world setting.

Hours logs

Date	# Hours	Description of work
12.01.2025	1	I spent time brainstorming the main idea for my project by reviewing my past work to analyze its strengths and identify areas for improvement. I focused on finding a completely new scope of work that could add unique value and elevate the project's impact.
17.01.2025	2	I wrote a project proposal and conducted research to establish the domain, define the problem. Created project contract.
20.01.2025	3	Explore existing literature to identify knowledge gaps. This included developing hypotheses, outlining the research design, methodology, technologies, and expected results, emphasizing how the findings could address practical challenges and contribute to the field. Created Repo on GitHub.
21.01.2025	2	Reading different articles, watching YouTube videos about regarding AI and chatbot implementation.
23.01.2025	1,5	I wrote a project proposal (section project planning and timeline). Creating Gantt Chart.
03.02.2025	2	I started by creating my Spring Boot project and setting up the pom.xml file, ensuring the correct dependencies were included for the initial setup. I added a welcome.html page as the main landing page and configured a HomeController to serve it properly. Along the way, I encountered SQL database connection issues, which I resolved by correctly configuring application.properties with the necessary MySQL setting. I implemented SecurityConfig.java, allowing public access to specific pages so that I wouldn't need to log in every time. Additionally, I had issues displaying images, which I resolved by placing them in the static/images/ directory and updating the image paths in the HTML.
04.02.2025	1	I worked on integrating my IntelliJ IDEA project with an existing GitHub repository through the desktop version, ensuring proper Git initialization, remote repository linking, and successfully committing and pushing my code.

05.02.2025	1.5	<p>I worked on refining the user registration process for the login page in my application. After users click "Register" on the login page, they are redirected to a user type selection page. I configured Spring Security to allow access to the /select-user-type endpoint, ensuring that users can view this page without needing to authenticate. Additionally, I updated the SecurityConfig to manage redirections correctly and resolved an HTTP 403 (Forbidden) error by adjusting the necessary permissions. I also verified that the "Go Back" button on the selection page accurately redirects users back to the login page.</p>
07.02.2025	2.5	<p>I worked on setting up the user registration flow for the NativeSpark project. Currently, I am implementing a multi-step registration process where users first select their type: Business, Indigenous Entrepreneur, or Basic User. For business users, I have designed a two-step process. In Step 1, users provide their email, password, and user type, which are stored in the users table. In Step 2, users enter their business details and upload a logo, which is saved in the business_users table.</p> <p>I have set up the necessary entities (User and BusinessUser), as well as repositories, services, and controllers to manage data persistence and form submission. Additionally, I created Thymeleaf templates for the registration pages.</p> <p>I also spent time debugging issues related to database mapping. Despite these efforts, saving BusinessUser data in the database is still not functioning correctly, and I will need to continue troubleshooting this issue in the next session. I also plan to implement a similar process for the other two user types.</p>
09.02.2025	3	<p>I focused on debugging the issue where user data wasn't saved in the database after Step 2 of registration. Initially, there were errors with the @JoinColumn reference in the BusinessUser entity. After fixing this, I encountered another problem: uploaded logo files were not being saved correctly, leading to an error. I updated the BusinessUserService to ensure the upload directory exists before saving files. The application was trying to save files in a non-existent directory within the Tomcat temporary workspace. I modified the code to create the directory if it didn't exist and changed the file-saving logic to use 'Files.copy()', ensuring proper storage of uploaded files.</p> <p>I implemented the Entrepreneur User Registration process, following the same structure as the Business User Registration. Similar to the Business Registration, the entrepreneur registration consists of two steps.</p>

		<p>In Step 1, the user enters their email and password, which are stored in the users table with the user_type set to "ENTREPRENEUR." In Step 2, the user provides personal details, including their first name, last name, a brief description about themselves, indigenous identity, and a profile photo. This information is saved in the entrepreneur_users table and is linked to the corresponding user.</p> <p>Writing logs and progress report 1.</p>
		<p>I implemented the Basic User Registration process, following the same structure as the Entrepreneur User Registration. Similar to the Entrepreneur Registration, the basic registration consists of two steps. In Step 1, the user enters their email and password, which are stored in the users table with the user_type set to "BASIC." In Step 2, the user provides personal details, including their first name, last name, a brief description about themselves, and a profile photo. This information is saved in the basic_users table and is linked to the corresponding user. This time without debugging issues.</p> <p>Initially, passwords were stored in plain text, but I needed to implement encryption using a PasswordEncoder. I updated the UserService class to include password encoding before saving user data to the database. However, I encountered an issue because UserService implemented UserDetailsService but did not override the `loadUserByUsername(String username)` method. This omission caused an error during the authentication setup in SecurityConfig.java.</p> <p>To resolve this, I implemented the missing `loadUserByUsername` method in UserService, ensuring that it retrieves user data from the database and returns a UserDetails object. This change enabled authentication to function properly with Spring Security. Additionally, I updated the authentication manager in SecurityConfig to utilize UserService for user authentication and to encode passwords before validation.</p>
11.02.2025	2.5	Some modifications in htmls files.
12.02.2025	1	Today, I added a confirmation message for successful registration. However, I encountered an issue where the data from Step 2 of the business registration process was not being saved to the database. After debugging, I discovered that the problem was caused by the `event.preventDefault();` call in the form submission, which was preventing the form from sending data to the backend. To resolve this issue, I removed the `onsubmit` event. Once I verified that the business registration flow was working correctly, I

		<p>implemented the same functionality for the Entrepreneur and Basic User registration processes to ensure consistency across all user types.</p> <p>New page subscription, need to connect it to all users.</p>
13.02.2025	2.5	Styling home page. I was trying to implement subscription as step 3 of registration, but half way through decided to leave for future account settings.
14.02.2025	2	Setting up the logging for users. Need to start again.
16.02.2025	3	Trial number two to set up login for users, unfortunately broke whatever had before.
23.02.2025	3	<p>I have started from scratch developing the login access to the account for users, the program logs in via users table.</p> <p>At this trial I also had some issues, my login wasn't working due to a conflict between the Authentication class from Apache Tomcat and the one from Spring Security. This conflict caused the authentication check to fail, preventing the user session from being recognized. Consequently, after logging in, the system redirected me back to the login page instead of taking me to the account page. To resolve this issue, I removed the incorrect import (<code>org.apache.tomcat.util.net.openssl.ciphers.Authentication</code>), ensured I was using <code>org.springframework.security.core.Authentication</code>, and updated my AccountController to properly verify if a user was authenticated using Spring Security.</p>
23.02.2025	2	Writing Midterm Report, How to Run The Program document, recording Video and etc.
03.03.2025	2.75	<p>After logging into my account, I focused on displaying user-specific details based on different user types: Basic, Business, and Entrepreneur. Initially, I could only see the email and user type, but additional details like names, descriptions, and logos were absent.</p> <p>The issue stemmed from my AccountController, which was retrieving user-specific entities (e.g., <code>BusinessUser</code>, <code>EntrepreneurUser</code>, etc.), but the data was not being correctly passed to the Thymeleaf template. After debugging, I ensured that the correct user details were loaded and properly mapped. As a result, I was able to fix the missing information.</p> <p>However, the logo for business users still wasn't showing because the uploaded images were stored in the uploads directory rather than in the static directory, which prevented Thymeleaf from accessing them directly. I</p>

		<p>adjusted the image path handling to serve the uploaded images properly, and after that, everything displayed correctly.</p> <p>Additionally, I implemented a logout feature for all users. While users are logged in, they can navigate within the platform, including the "My Account" page, but once they log out, they cannot access the "My Account" page anymore.</p>
04.03.2025	2.67	<p>Today, I worked on implementing the subscription change feature within the user accounts of my web application. When users register, they receive a Free subscription by default. I wanted to give them the option to upgrade or downgrade their plan from their account settings.</p> <p>Initially, I encountered an issue where the subscription wasn't updating in the database. I resolved this by ensuring the backend correctly retrieved the authenticated user and updated their subscription information in the database.</p> <p>Next, I faced a "bootstrap is not defined" error when trying to display the confirmation modal. This occurred because Bootstrap's JavaScript file wasn't loaded properly. I fixed this by adding the correct Bootstrap JS file just before the closing </body> tag.</p> <p>After that, I encountered another problem: the frontend failed with the error "Unexpected token 'S'" when saving the subscription. This happened because the backend was returning a plain text response instead of JSON. To fix this, I made the backend send a JSON response and updated the frontend to properly parse it.</p>
05.03.2025	2.67	<p>Today, I began by separating the controller logic for each user type: Business, Entrepreneur, and Basic. This allowed me to test and manually update the HTML form, ensuring that user data was being correctly updated. Initially, I entered the update paths manually in the HTML form for each user type to verify the data submission process.</p> <p>After confirming that the updates worked for each type individually, I combined all three user types into a single controller. This new setup would handle updates dynamically based on the logged-in user's type.</p> <p>However, I encountered an issue where the "Save" button did not appear when clicking "Edit," and the file input for logos/photos was not enabling for Entrepreneur and Basic users. I resolved this by adjusting the JavaScript to check the user type before enabling the appropriate fields, ensuring that only the relevant inputs were editable for each user type.</p>

06.03.2025	1.88	<p>I was working on enabling file uploads so that Business users could upload a logo, while Entrepreneur and Basic users could upload a profile photo. I also needed to ensure that these updates reflected on the account page.</p> <p>The main issue I encountered was that the file input fields were not activating correctly for different user types, and the uploaded images were not being saved to the database properly. I resolved this by adjusting the JavaScript `toggleEdit()` function to enable the correct input based on user type. Additionally, I updated the form to include both logo and photo inputs, and I modified the backend services and controller to correctly handle file uploads, saving them in the appropriate directories.</p> <p>I worked on implementing a transaction system to log subscription changes when users upgrade or downgrade their plans. Previously, changing a subscription only updated the Subscription table without retaining a record of past changes. To address this, I added a Transactions table to store the subscription type, cost, and timestamp whenever a user updates their plan.</p> <p>One challenge I faced was ensuring that transactions were recorded only for paid plans (Basic and Premium) while treating "Free" as a non-billable change. I resolved this by incorporating conditional logic in the backend to log only the relevant changes.</p> <p>Additionally, I tackled a UI issue where canceled profile edits still displayed modified (but unsaved) data. To fix this, I implemented a Cancel button that refreshes the page to reload the original database values. As a result, both subscription changes and profile edits now function smoothly, ensuring data accuracy and proper tracking.</p>
10.03.2025	2.87	<p>Renamed transactions table to orders for future differentiation of transactions for the business and the customers. Also had an error when the table was named order.</p> <p>The error occurs because order is a reserved keyword in SQL. Since order is used for ordering results in queries, using it as a table name causes a syntax error.</p> <p>Today, I established the posting functionality for different user types. While the postings are not yet visible on the front end, I have successfully created the necessary tables in the database: job_postings, project_postings, and products. The controller is set up, allowing business users to manage job and project postings, while entrepreneur users can list products for sale. This</p>

		<p>functionality is role-specific, ensuring that businesses can add job and project postings, whereas entrepreneurs can add products along with images.</p> <p>I also implemented a toggle switch for business users, allowing them to easily switch between job and project postings while ensuring that the correct data is displayed. Additionally, I added a "Back to Account" button to enhance navigation. Access to "My Postings" has been restricted, so Basic users cannot see this option in the navigation menu.</p> <p>During this process, I encountered security role issues and repository mismatches, which I resolved by refining security configurations and adjusting database mappings.</p>
11.03.2025	2.53	<p>Today, I worked extensively on refining the functionalities for posting jobs, projects, and products in my Spring Boot application. I began by ensuring that business users can create job and project postings, while entrepreneurs can list their products for sale. I organized my Thymeleaf templates to properly display these listings, making sure that the images for product postings appear correctly and are aligned to the left.</p> <p>I encountered some issues during this process, so I improved the image upload logic for products. This ensures that uploaded images are stored correctly and displayed on the "My Postings" page.</p>
12.03.2025	2.58	<p>Today, I worked on implementing the delete functionality for job postings, project postings, and product postings. Initially, I encountered a "405 Method Not Allowed" error when attempting to delete a posting. This issue arose because the delete request was being blocked by CSRF protection in Spring Security.</p> <p>Additionally, using <code>@DeleteMapping</code> directly in the controller did not function properly with Thymeleaf forms, which default to sending POST requests. To resolve this, I modified the Thymeleaf delete buttons to include a hidden input field with <code>_method=DELETE</code>. This change allowed the form to properly send a DELETE request while still adhering to CSRF protection.</p> <p>I also updated the controllers to use <code>@PostMapping</code> for the delete endpoints instead of <code>@DeleteMapping</code>, ensuring that requests were correctly processed.</p> <p>I have implemented the Edit Posting functionality, making updates to both the frontend and backend of my application.</p> <p>On the frontend, I created an HTML page that maintains consistent styling with the job and project editing pages. Additionally, I added an image upload</p>

		<p>field, allowing users to update their product image while retaining the existing one if no new file is provided.</p> <p>On the backend, I made several important modifications to the ProductPostingController to support the editing feature. I introduced a GET mapping to load the existing product data into the form and a POST mapping to handle updates. The controller retrieves the product by its ID, ensuring that only authenticated entrepreneurs can modify their own postings. I also implemented logic to manage image file updates, so if a new image is uploaded, the old one is replaced; otherwise, the previous image remains unchanged.</p>
13.03.2025	2.92	<p>Today, I refined the home page functionality to ensure different types of users, including non-logged-in visitors, see the appropriate content. I solved an issue where unauthenticated users were wrongly redirected to the login page by adjusting the controller logic to keep the product list accessible to everyone.</p> <p>I also addressed a problem with product images not displaying for non-logged-in users due to incorrect image path references. I fixed this by updating the Thymeleaf syntax to properly serve images from the static resources directory.</p> <p>For entrepreneur users, I implemented a toggle bar to switch between products, job postings, and project postings. The challenge was keeping the clicked button visually active (highlighted in blue). I resolved this by using JavaScript to toggle Bootstrap classes, ensuring only the active button remained highlighted.</p> <p>Moreover, I fixed internal server errors (500) that occurred when entrepreneur users clicked "Home" in the navbar by updating the controllers to ensure the correct data was fetched for different user roles. Lastly, I improved the navbar behavior so that clicking "Home" directs users to the correct page based on their authentication status and role. Entrepreneurs can now see products, jobs, and projects, while basic users can only view products. These changes greatly enhanced the user experience and navigation consistency.</p> <p>I encountered an issue where removing some commented-out code caused the home button in the navbar to malfunction for unauthenticated users, redirecting them to the login page instead of displaying products. This was confusing since the home page should be accessible to everyone.</p>

		<p>Upon investigation, I found that the commented-out code contained crucial configurations that affected user authentication behavior. To fix the issue, I made sure products were visible to all users, allowing both authenticated and unauthenticated individuals to browse without needing to log in.</p>
17.03.2025	3.067	<p>I implemented a detailed view for all listings, allowing users to access details for products, jobs, and projects with a click. Now, when a logged-in user clicks a listing, it correctly opens the detailed page. In contrast, an unauthenticated user is redirected to the login page before they can access any listing details.</p> <p>I also improved the marketplace page for all users to ensure a smoother experience when navigating between products, jobs, and projects. Previously, clicking the toggle buttons redirected users to the product details page because of the stretched-link class, which made the entire card clickable, including the buttons. To resolve this, I modified the clickable areas so that only the title or image of each listing is clickable.</p> <p>I was attempting to implement a "like" functionality that would allow users to like a product, with the number of likes being stored and updated in the database. However, despite numerous attempts to debug and fix the code, the functionality is not working as expected. The likes are not being saved in the database, and the user interface does not update correctly when a user clicks the like button. Even after verifying that the repository, controller, and JavaScript functions were properly configured, the issue still persists. At this point, I have decided to start from scratch and rebuild the like feature from the ground up to ensure it functions correctly.</p>
19.03.2025	2.75	<p>Today, I worked on implementing the purchase functionality for my project. I created a purchase page where users can buy products from entrepreneurs. This page includes seller information, product details, and a purchase form for users to enter their shipping and payment details. I ensured that the form properly handles user input, including selecting the quantity, filling in address fields, and adding payment information. Additionally, I integrated the form with the backend so that when a user submits it, the data is sent correctly.</p> <p>On the product details page, I added a buy button for all users and a "request custom amount" option specifically for business users. If the quantity of a product is zero, the buy button becomes hidden. I also corrected an issue on the homepage where the user who posted a product could see their own listing, which should not be the case.</p>

		<p>On the backend, I set up the PurchaseController to handle form submissions and process orders. I ensured that the data is saved correctly in the database and that the appropriate associations are made between the user, product, and seller. During this process, I debugged several issues related to missing fields and incorrect mappings in my entities. At one point, I encountered a SQL error because a required field was missing a default value, but I resolved it by updating my entity and database schema.</p>
20.03.2025	2.73	<p>Today, I developed a page that allows users to view their product orders and enables sellers to see who purchased their products. I created separate views for buyers and sellers. Users can access a "My Orders" page that displays a list of all the products they've purchased, including details such as quantity, product information, and shipping data. Sellers can access a "Buyers of My Products" page, which shows who bought each of their listed products. On the backend, I implemented the necessary methods in the "OrderController" to fetch the relevant transaction data based on the logged-in user's role.</p> <p>I also worked on implementing and debugging the save/unsaved feature for products, job postings, and project postings on the NativeSpark platform. I built the logic for the save button using a toggle mechanism that updates the database through POST requests to endpoints such as /product/{id}/toggle-save, /job/{id}/toggle-save, and /project/{id}/toggle-save. Each listing entity has a savedByUsers field that maps a many-to-many relationship with the User entity, creating new tables in the database. This setup allows multiple users to save the same listing and enables users to manage their saved items.</p> <p>On the frontend, I added heart icons next to each listing card, which change color when clicked—red when saved and white when unsaved. The initial implementation visually toggled the heart correctly on click; however, I noticed that after reloading the page or navigating away and back to the homepage, the heart icon did not accurately reflect whether an item was saved.</p> <p>To address this issue, I improved the toggleSave function in the JavaScript code to read the server's text response (saved or unsaved) and update the heart icon to display ❤️ or 🚫 accordingly. This enhancement significantly improved the visual feedback after saving or unsaving an item.</p>
21.03.2025	1.55	<p>Today, I added the CustomProductRequest entity, along with the CustomProductRequestController, CustomProductRequestRepository, and the corresponding request_custom_amount.html page. This addition allows business users to request a custom amount of a product, and these requests</p>

		<p>are now saved in the database. I also included a success message on the page and added a "Back to Home" button to enhance the user experience.</p> <p>In addition, I worked on feedback to better distinguish job postings from project postings. I removed the "Contract" option from job postings to prevent any overlap. The feedback also emphasized the importance of fully implementing individual features before expanding them. As a result, I plan to revise the product purchase flow. Instead of allowing users to buy products directly, they will now be able to add products to a cart and purchase from there, making the functionality more complete and user-friendly.</p>
25.03.2025	3	<p>I implemented full cart functionality for authenticated users. First, I created the `Cart` and `CartItem` entity classes. Each cart is linked one-to-one with a user, and a cart can contain multiple items. Each `CartItem` stores information about the product and its quantity.</p> <p>I added a cart field to the `User` entity and updated the `UserService` so that when a user registers, a new cart is automatically created and linked to them. Next, I created `CartRepository` and `CartItemRepository` to manage the carts and their items in the database.</p> <p>In the `CartController`, I added a `/cart` endpoint to load the user's cart and display it using Thymeleaf. Additionally, I implemented a `/cart/add` POST endpoint to handle adding products to the cart. This endpoint checks if the item already exists; if it does, it updates the quantity; if not, it creates a new cart item. I also added the necessary getters and setters for quantity in the `CartItem` class.</p> <p>On the frontend, I updated the `product-details.html` page to include a form that allows users to select a quantity and submit their choice. The cart page displays all cart items in a table, showing the product name, description, price, quantity, and total. It also manages scenarios where the cart is empty.</p>
26.03.2025	2.5	<p>In the CartController, I replaced the old `addToCart` method, which used form parameters, with a new version that accepts a `CartItemRequest` object in JSON format. This change facilitates smoother interactions on the frontend without requiring full-page reloads. I also implemented proper HTTP responses using `ResponseEntity` to return appropriate status codes and messages.</p> <p>To support this functionality, I created a new Data Transfer Object (DTO) class called `CartItemRequest`, which contains fields for `productId` and</p>

`quantity`. Additionally, I added the missing getter and setter methods in the `CartItem` entity for both the cart and product.

On the frontend, I updated the product details page to handle cart addition using JavaScript's `fetch` method, sending data as JSON and displaying a Bootstrap toast notification to indicate success or error. Furthermore, I developed a `cart-checkout.html` page styled consistently with the rest of the app, which displays products added to the cart, including their name, price, quantity, and total.

I added a reusable navbar and footer across some pages to create a more polished and cohesive user experience.

I improved the cart functionality by implementing a complete checkout flow. In the CartController, I added a `/cart/summary` endpoint to show the cart summary with total quantity and price, along with a `cart-summary.html` view for a clean presentation.

The Cart entity now features a `List<CartItem>` with appropriate JPA annotations, enabling access to all items in the user's cart. I also created endpoints for updating item quantities and removing items at `/cart/updateQuantity` and `/cart/removeItem`.

On the frontend, I developed a styled `cart.html` page listing all items with forms for updates and removals, linked to a checkout summary via a "Checkout" button. The `cart-checkout.html` page collects shipping and payment information using a structured form while maintaining a consistent navbar and footer across all views.

In the CartController, I added GET and POST handlers for the `/cart/checkout` route. The `getCheckout()` method returns the `cart-checkout.html` view, and the `completeCheckout()` method processes the form submission, capturing shipping and payment details after verifying user authentication.

I saved a new Transaction post-purchase by building a Transaction object with shipping details and purchase information. After marking the status as "SUCCESS" and saving to the database, I passed the `transactionId` and `totalPrice` to the success page.

Additionally, I updated entity relationships by linking Product to transactions with a `@ManyToMany` association and created a join table called `transaction_product`. The Transaction entity now supports multiple products, laying the groundwork for future multi-item checkouts.

27.03.2025	2	<p>I implemented a basic product recommendation system and developed the data seeding infrastructure to support it. I created a ` `.env` file and utilized the ` `dotenv` package to securely load the database credentials into the application. In the ` `requirements.txt` file, I listed the necessary libraries, including ` `mysql-connector-python` , ` `pandas` , ` `SQLAlchemy` , and ` `scikit-learn` .</p> <p>I added ` `alchemy.py` to manage SQLAlchemy-based connections and to facilitate table reflection for easier Object-Relational Mapping (ORM) access to existing MySQL tables. Additionally, I created ` `basic.py` to implement the recommendation logic, which uses cosine similarity based on product categories and normalized prices, storing the results in a new recommendations table.</p> <p>Furthermore, I developed ` `seed.py` to generate realistic fake data using the ` `Faker` library for users, entrepreneur profiles, products, and thousands of transactions. This script ensures relational integrity, generates consistent random data, and populates the MySQL database cleanly through ` `cursor.execute()` and ` `commit()` . This setup lays the groundwork for building personalized recommendation features within the application.</p> <p>I implemented a system to display personalized product recommendations directly on the product details page. To achieve this, I created a Recommendation entity to define the relationships between a product and its recommended products. This involved using two relationships: one for the main product and another for the recommended product. In the Product entity, I added a relationship to connect each product with its list of recommendations.</p> <p>On the frontend, I updated the product details page to include a new "Recommendations" section located below the main product information. This section loops through the recommendations using ` `product.getRecommendations()` and displays each recommended product as a card, showcasing its image, name, category, and a link to its detail page.</p> <p>There is also more advanced file for recomedations but it requires a lot of purchase history in transactions table. I will stick to basic recomendations but category and the price.</p>
30.03.2025	2	The python script wasn't working, and the random generation in the database user and products.

		<p>I have no idea what happened, had to reboot computer few times and created a list of commands that made everything work:</p> <pre> cd desktop cd W25_4495_S2_ValeriiaN cd recommendations Set-ExecutionPolicy -ExecutionPolicy Unrestricted -Scope CurrentUser venv/Scripts/activate pip install -r requirements cd src python basic.py python -m db.seed python basic.py Writing progress report. </pre>
02.04.2025	2.5	<p>Today, I spent time testing the program and improving the structure and functionality across the various pages. I separated the navbar and footer into individual Thymeleaf fragment files to reduce code repetition and successfully connected them to each page using 'th:replace'. I also resolved some issues with Bootstrap by ensuring the correct scripts were properly loaded, which fixed the dropdown menu that was not functioning on certain pages. Debugging and testing this took quite a while across different views. As the program expands, it takes more time to test all of its functionality. I have been trying to separate it into different blocks to ensure that changes in one block do not affect the others, but it is not working that way al the time.</p> <p>During testing, I encountered a few issues. The "Buyers of My Products" page in the entrepreneur account is not displaying data due to recent changes made in the database schema. I attempted to fix this, but it is still not working as expected. Additionally, I discovered that when a user purchases a product, the product quantity is not updating in the database. This issue needs to be addressed to maintain accurate stock levels.</p>
03.04.2025	3	<p>Today, I fixed the buyers' page, which had been crashing due to a change in the database schema. The transaction entity no longer had a direct reference to the product, causing an error in the Thymeleaf template that attempted to access `sale.product.productName`, which no longer existed. To resolve this</p>

		<p>issue, I updated the page to loop through `transactionProducts` and accessed the product information from there.</p> <p>Additionally, I addressed some styling problems on the user registration page. Yesterday, I accidentally removed Bootstrap, but I reintroduced it today and cleaned up the layout.</p> <p>Fixed the product qty update after successful purchase.</p> <p>Today, I encountered an issue while trying to delete a product. I received a foreign key constraint error because the product was still referenced in the recommendations table as a recommended product. Essentially, Hibernate couldn't delete it since other rows were still pointing to it.</p> <p>To resolve this, I realized that I had only mapped the recommendations list for the products that recommend others and not the inverse — the products that are being recommended. So, I added a new list called `recommendedBy` in the Product entity with `mappedBy = "recommendedProduct"` and used `cascade = CascadeType.ALL` along with `orphanRemoval = true`.</p> <p>After making these changes, deleting the product worked perfectly, and all related recommendations were removed automatically.</p> <p>Testing the entire thing....</p>
04.04.2025	1.67	<p>Today, I worked on the custom product request feature. I updated the product details page so that when the quantity is 0, there is now a "Request Custom Amount" button visible for all users, including both Entrepreneur and Basic users, not just Business users. Additionally, I built the "My Requests" page to list all custom requests a user has made. I fixed some issues with the controller mapping, which previously returned a 404 error, and added the "My Requests" link into the services dropdown in the navbar. To ensure it displayed correctly, I had to change its class to "dropdown-item."</p> <p>Also, I have updated the design of the request custom amount page.</p>
04.04.2025	0.67	<p>I worked on a feature that allows entrepreneurs to view a list of users who submitted custom product requests. I created a page that iterates through each CustomProductRequest and displays the product name, requested amount, request time, and the requester's information. Initially, I attempted to show the requester's full name using `req.user.firstName + '' + req.user.lastName`, but the application crashed with a 500 Internal Server Error. Upon reviewing the User entity, I discovered that it does not contain fields for first name or last name; it only has an email field. To resolve this issue, I updated the template to display just the email instead: `<span</p>

		<code>th:text="\${req.user.email}">`.</code> This adjustment resolved the problem, and the requests now load correctly on the page.
06.04.2025	1.83	<p>Today, I implemented search functionality on the home page that enables users to search for products using keywords found in the name or description. Additionally, I expanded the search feature for entrepreneurs to include job and project postings based on keywords found in the descriptions and required skills. I also introduced filtering options: for products, users can filter by category, which dynamically loads from the database, while for jobs, entrepreneurs can filter by employment type (Full-time, Part-time, Internship). These search inputs and filters were integrated into the existing home page form and connected to the controller logic to apply the filtering when rendering the data.</p> <p>Creating questions for basic users for the survey:</p> <ol style="list-style-type: none"> 1. How easy was it to navigate the platform and find what you were looking for? 2. Did the product listings (images, descriptions, pricing) provide enough information for you to make a decision? 3. How likely are you to purchase Indigenous-made products through NativeSpark? 4. What features do you find most valuable as a basic user? 5. If a product is out of stock, how useful is the 'Request Custom Amount' feature for you? 6. What additional features would make you more likely to use this platform regularly? 7. How comfortable were you with the design and layout of the site? 8. Did you experience any issues while browsing, searching, or saving items? 9. Based on what you've seen and the idea presented, how likely are you to recommend NativeSpark to others?
09.04.2025	2	<p>Writing final report.</p> <p>Updated the Installation Instructions.</p>
10.04.2025	0.5	Explaining the idea to my basic users and business to collect the feedback
10.04.2025	1.75	User Guide report

11.04.2025	4	Final Report Presentation Speech Deck
13.04.2025	1.5	The recommendation script was failing because I used the scaler object without defining it inside the function where it was needed. I fixed it by moving scaler = MinMaxScaler() inside the loop to ensure it's properly initialized for each category subset. Populating the program a little bit.

References

1. Statistics Canada. (2023, July 18). Indigenous entrepreneurship in Canada. Retrieved from <https://www150.statcan.gc.ca/n1/daily-quotidien/230718/dq230718c-eng.htm>
2. Richmond, C. A. M., & Cook, C. (2017). Creating conditions for Canadian Aboriginal health equity: The promise of healthy public policy. *Social Science & Medicine*, 176, 93–103.
3. Indigenous Corporate Training Inc. (n.d.). 8 key issues for Indigenous peoples in Canada. Retrieved from <https://www.ictinc.ca/blog/8-key-issues-for-indigenous-peoples-in-canada>
4. Indigenous Industrial and Contracting Network. (n.d.). Retrieved from <https://www.imcn.ca/>
5. Indigenous Business Development Services. (n.d.). Retrieved from <https://ibdssk.com/>
6. Python Software Foundation. (n.d.). *venv — Creation of virtual environments*. Python 3 documentation. Retrieved April 9, 2025, from <https://docs.python.org/3/library/venv.html>

Tools used

AI Tools were used for coding backend and frontend, ML Scripts, reports (for proper English, Grammarly to polish things off).

7. W3Schools. (n.d.). *W3Schools Online Web Tutorials*. Retrieved April 9, 2025, from <https://www.w3schools.com/> (Coding)
8. OpenAI. (2023). ChatGPT (Mar 14 version) [Large language model]. <https://chat.openai.com/chat> (Paid, used for coding backend and frontend, ML Scripts, debugging, Reports)
9. Grammarly. (n.d.). Grammarly. Retrieved April 9, 2025, from <https://www.grammarly.com/> (Reports)
10. DeepSeek. (n.d.). *DeepSeek: Exploring Uncharted Territories*. Retrieved April 9, 2025, from <https://www.deepseek.com/> (used for coding backend and frontend)
11. Canva. (n.d.). *Amazingly Simple Graphic Design Software – Canva*. Retrieved April 9, 2025, from <https://www.canva.com/> (Paid, Presentation)

Downloads links:

12. Python Software Foundation. (n.d.). *Download Python*. Retrieved April 9, 2025, from <https://www.python.org/downloads/>
13. Oracle Corporation. (n.d.). *MySQL: The world's most popular open source database*. Retrieved April 9, 2025, from <https://www.mysql.com/>
14. JetBrains. (n.d.). *IntelliJ IDEA: The IDE for Pro Java and Kotlin Development*. Retrieved April 9, 2025, from <https://www.jetbrains.com/idea/> (Douglas Credentials)
15. Apache Friends. (n.d.). *Apache Friends - XAMPP*. Retrieved April 9, 2025, from <https://www.apachefriends.org/>

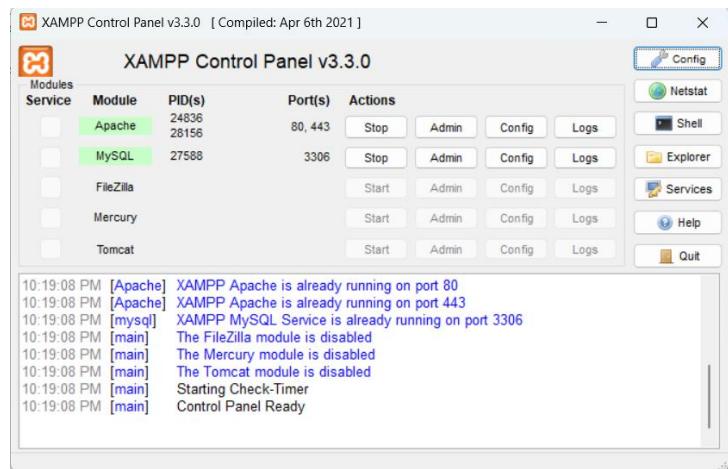
Other

16. Friends help

Appendix A: Installation Guide

1. Start XAMPP:

- Open the XAMPP control panel.
- Start the **Apache** and **MySQL** services.



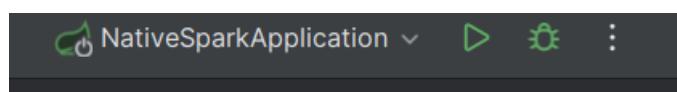
-

2. Open the NativeSpark Project:

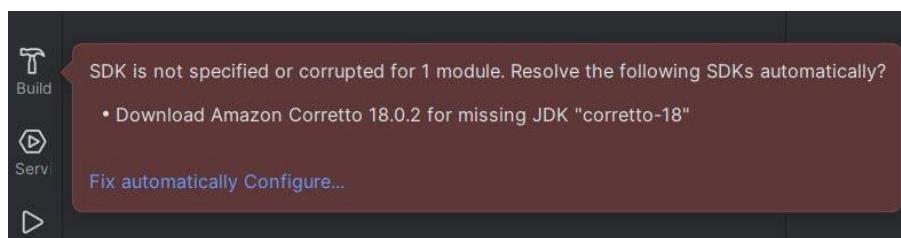
- Launch **IntelliJ IDEA**.
- Open the NativeSpark project

3. Run the Application:

- Click the **Run** button in IntelliJ IDEA to start the application.



- In case of the warning like below, please press “Fix auto”



4. Run the Python Script:

- Open the Terminal and run the following commands:
- `cd desktop cd W25_4495_S2_ValeriiaN`

- cd reccomendations Set-ExecutionPolicy -ExecutionPolicy Unrestricted -Scope CurrentUser
- venv/Scripts/activate
- pip install -r requirements
- cd src
- python basic.py
- python -m db.seed (if needed to populate the database with fake data)
- python basic.py (repeat the following command to update the product recommendations based on the added products to the database)

5. Access the Application:

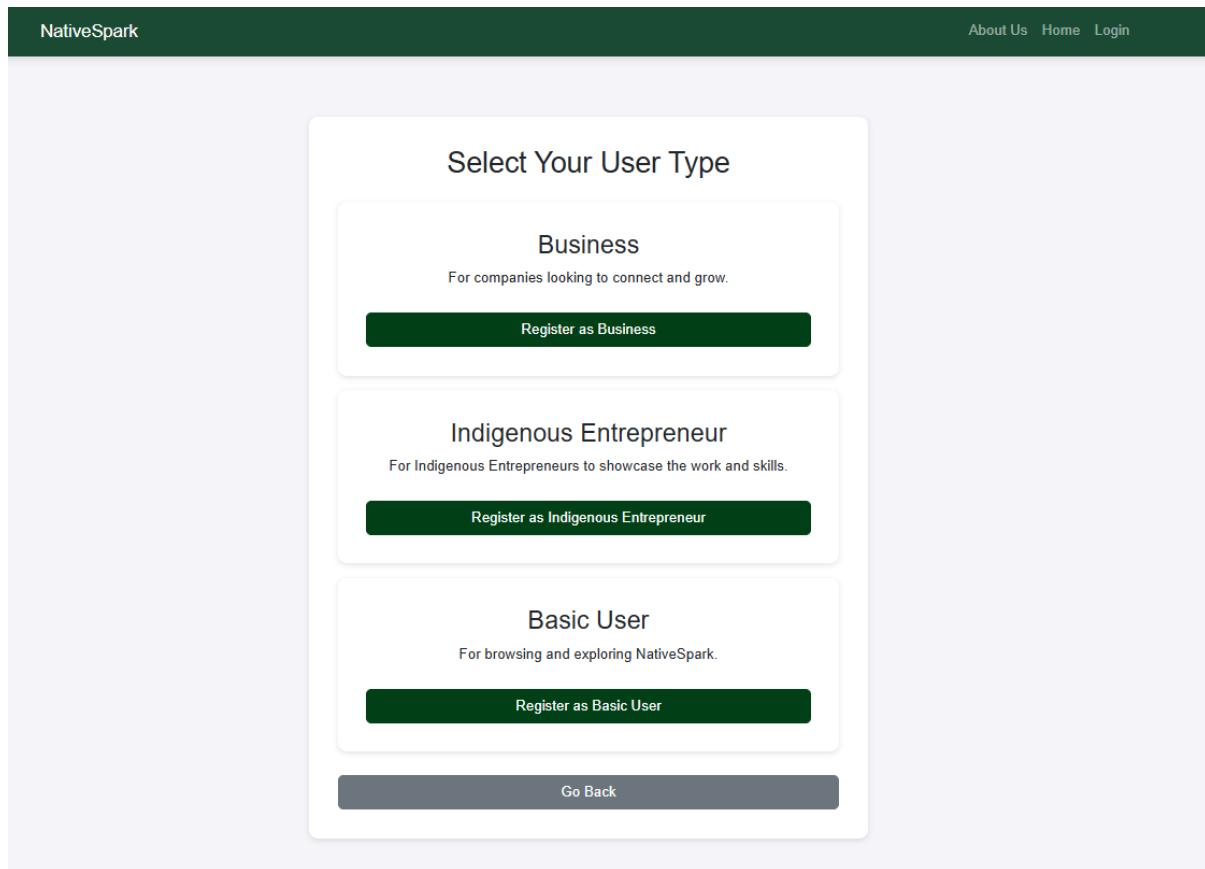
- Open a web browser.
- Navigate to <http://localhost:8082> to access the NativeSpark application.

 localhost:8082/about

Appendix B: User Guide

Registration

The user must select a role before accessing the application and registering.



The first step for all three users is the same: to choose the login (email) and password. The second step asks for specific information addressed to the user type.

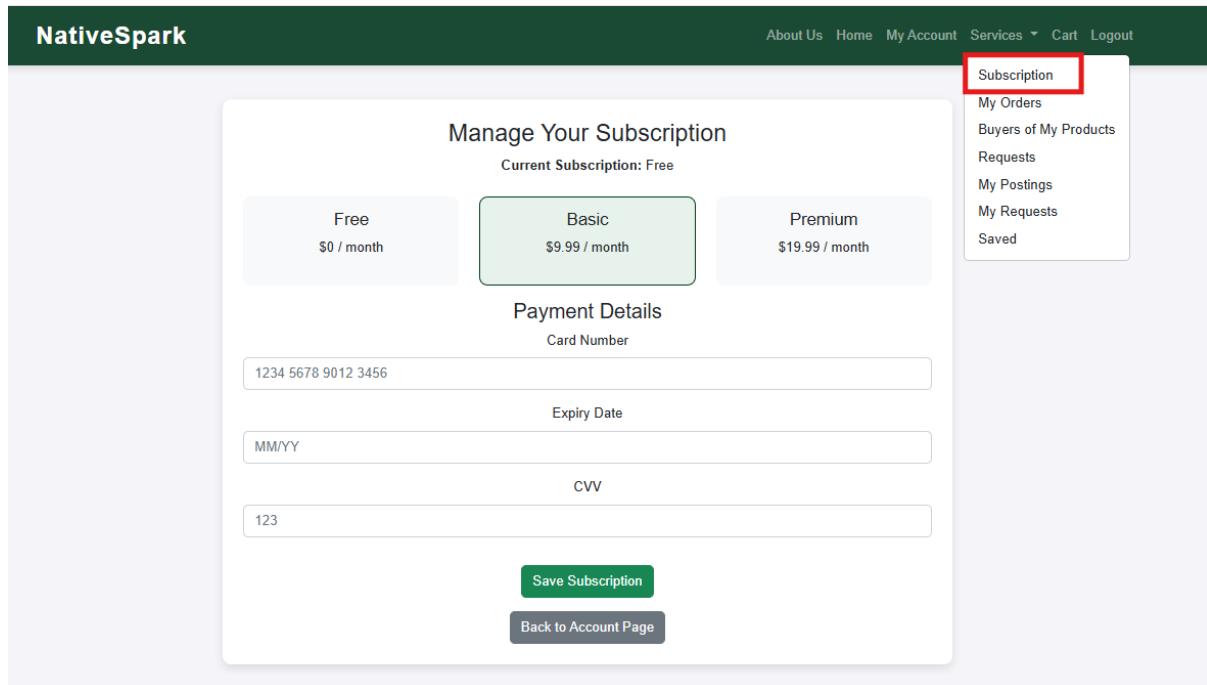
Business Registration - Step 1

[Next →](#)

[← Back to User Type Selection](#)

Subscription

Thee uses can change their subscription Service dropdown



NativeSpark

- About Us
- Home
- My Account
- Services ▾
- Cart
- Logout

Manage Your Subscription

Current Subscription: Free

Free
\$0 / month

Basic
\$9.99 / month

Premium
\$19.99 / month

Payment Details

Card Number

1234 5678 9012 3456

Expiry Date

MM/YY

CVV

123

[Save Subscription](#)

[Back to Account Page](#)

- Subscription
- My Orders
- Buyers of My Products
- Requests
- My Postings
- My Requests
- Saved

Home Page

Non-logged-in users can browse products posted by Indigenous entrepreneurs on the home page. To view product details, users must log in. They can also search for or filter products.

The screenshot shows the NativeSpark Marketplace homepage. At the top, there is a dark green header bar with the "NativeSpark" logo on the left and navigation links "About Us", "Home" (which is highlighted with a red box), and "Login" on the right. Below the header, the page title "Marketplace" is centered above a search bar with placeholder text "Search jobs, projects, or products..." and a "Search" button. A "Products" section follows, featuring a dropdown menu "All Categories" and a "Filter" button. Three product cards are displayed in a grid:

- About Arm**
Category: Electronics
 Price: \$208.34
Description: Sister purpose six computer challenge. Billion fear spring note quickly still kitchen.
- Central Break**
Category: Clothing
 Price: \$89.45
Description: Defense believe man minute. Difference employee wonder executive outside region after network.
- Box Hope**
Category: Books
 Price: \$384.45

Business user

Registration

Business Registration - Step 2

Business Name

Business Description

Upload Logo

Choose File No file chosen

Register ✓

My Account

In my account you can edit information about yourself.

NativeSpark

About Us Home **My Account** Services Cart Logout



My Account

Email
b1@gmail.com

User Type
BUSINESS

Subscription
Free

Business Name
Anyvisa services corp.

Description
The best Company in the World.

Upload New Logo

Choose File No file chosen

Edit Profile **Save Changes** **Cancel**

© 2025 NATIVESPARK INTEGRATION SOLUTIONS CORP. All Rights Reserved.

Home Page

Business users on the home page can filter products by existing categories or by keywords searched in the product description and name.

The screenshot shows the NativeSpark Marketplace home page. At the top, there is a navigation bar with links: About Us, Home (which is highlighted with a red box), My Account, Services, Cart, and Logout. Below the navigation bar is a search bar labeled "Search jobs, projects, or products..." with a "Search" button. The main area is titled "Marketplace". On the left, there is a sidebar titled "Products" with a dropdown menu for "All Categories" showing options like Electronics, Clothing, Books, Sports, and Home. A "Filter" button is also present. The main content area displays two product cards: "Central Break" (Category: Clothing, Price: \$89.45) and "Box Hope" (Category: Books, Price: \$384.45). Each card includes a "Heart" icon for favoriting and a "Product Image" link.

Users can save their favourite products for later purchase or set a custom amount if their desired quantity exceeds the stock available at that moment.

This screenshot shows the same NativeSpark Marketplace home page as the previous one, but with a different set of products. The sidebar dropdown for "All Categories" now shows "About Arm" instead of the previous categories. The main content area displays two product cards: "About Arm" (Category: Electronics, Price: \$208.34) and "Central Break" (Category: Clothing, Price: \$89.45). Both cards feature a "Heart" icon with a red border, indicating they have been favorited. The rest of the interface is identical to the first screenshot, including the navigation bar and search bar.

Additionally, by clicking on the product image or name, a detailed summary will open.



The screenshot shows a product detail page for 'Lego'. At the top, there's a large image of the LEGO logo. Below it, the word 'Lego' is displayed in a bold, black font. To the right of the image, product details are listed: Category: Other, Price: \$151.0, Available Quantity: 100, Posted Date: 2025-04-04 00:03, and Description: LEGOs are plastic building-block toys that rose to massive popularity in the mid-20th century. A seller information section shows Pavel Sitnik. Below the details are buttons for quantity selection (1), 'Add to Cart', 'Request Custom Amount', and 'Back to Home'.

Recommendations

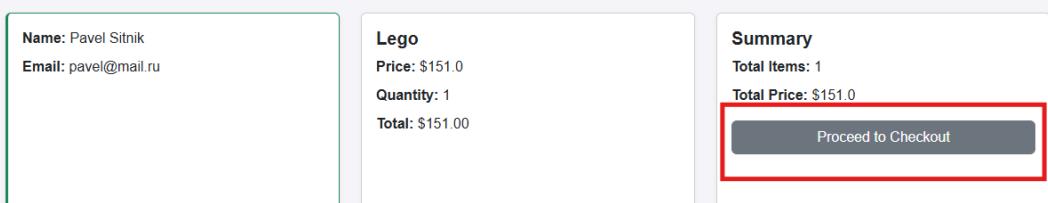
Cart

The product can be added to cart and proceeded to check out.



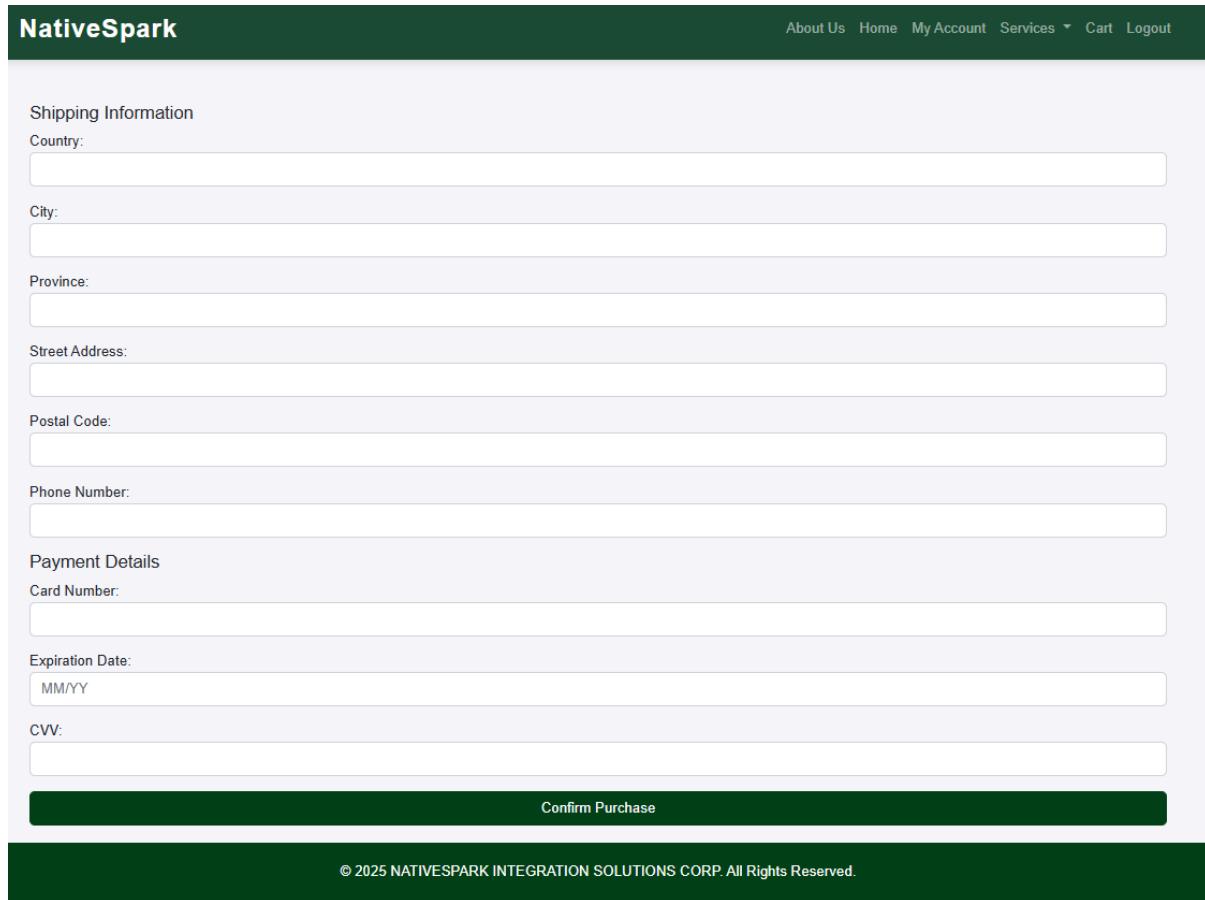
The screenshot shows a 'Your Cart' page. It features a table with columns: Product Name, Description, Price, Quantity, Total, and Actions. There is one item in the cart: 'Lego' with a description of 'LEGOS are plastic building-block toys that rose to massive popularity in the mid-20th century.' The price is \$151.0, quantity is 1, and total is \$151.0. Action buttons include 'Update' and 'Remove'. Below the table is a 'Proceed to Checkout' button.

On summary page the information about seller and the product will be shown.



The screenshot shows a summary page with three main sections: Seller Info, Product Info, and Summary. The Seller Info section contains the seller's name (Pavel Sitnik) and email (pavel@mail.ru). The Product Info section contains the product name (Lego), price (\$151.0), quantity (1), and total (\$151.00). The Summary section contains a 'Summary' heading, 'Total Items: 1', 'Total Price: \$151.0', and a 'Proceed to Checkout' button, which is highlighted with a red box.

To place the order shipping information needed.



The screenshot shows a form titled "Shipping Information" under the "NativeSpark" header. It contains fields for Country, City, Province, Street Address, Postal Code, and Phone Number. Below this is a "Payment Details" section with fields for Card Number, Expiration Date (MM/YY), and CVV. A "Confirm Purchase" button is at the bottom. The footer reads "© 2025 NATIVESPARK INTEGRATION SOLUTIONS CORP All Rights Reserved."

Shipping Information

Country:

City:

Province:

Street Address:

Postal Code:

Phone Number:

Payment Details

Card Number:

Expiration Date:

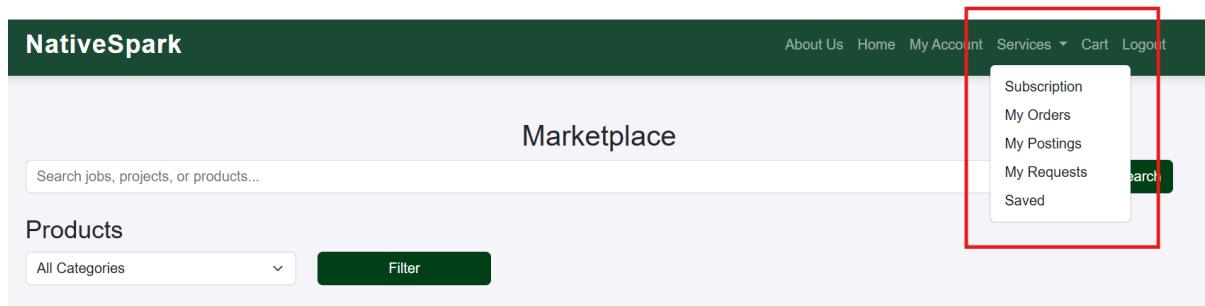
MM/YY

CVV:

Confirm Purchase

© 2025 NATIVESPARK INTEGRATION SOLUTIONS CORP All Rights Reserved.

Services dropdown



The screenshot shows the "Marketplace" section of the NativeSpark website. It includes a search bar, a "Products" section with a dropdown for categories and a "Filter" button, and a sidebar with a "Services" dropdown menu. The "Services" menu is open and highlighted with a red box, showing options like Subscription, My Orders, My Postings, My Requests, and Saved. A "Search" button is also visible in the sidebar.

NativeSpark

About Us Home My Account Services Cart Logout

Marketplace

Search jobs, projects, or products...

Products

All Categories Filter

Subscription
My Orders
My Postings
My Requests
Saved

Search

My Orders

To see the information about purchased products click My Orders

NativeSpark

About Us Home My Account Services ▾ Cart Logout

My Orders

Full Spend

Quantity: 2

Total Price: \$305.94

Seller: hillstephen@example.com

Phone Number: 45678

Status: SUCCESS

Shipping Address: 216 ltbleu, hjghfgzg, bc, canada, 5tghb7

Date: 2025-04-04 11:18

Lego

Quantity: 1

Total Price: \$151.00

Seller: pavel@mail.ru

Phone Number: 45678

Status: SUCCESS

Shipping Address: 216 ltbleu, hjghfgzg, bc, canada, 5tghb7

Date: 2025-04-04 11:18

Lego

Quantity: 2

Total Price: \$302.00

Seller: pavel@mail.ru

Phone Number: 45678

Status: SUCCESS

Shipping Address: 216 ltbleu, van, bc, canada, 5tghb7

Date: 2025-04-04 15:04

Lego

Quantity: 1

Total Price: \$151.00

Seller: pavel@mail.ru

Phone Number: 6045006272

Status: SUCCESS

My Requests

To request custom amount the user needs to submit it on product details page.

NativeSpark

About Us Home My Account Services Cart Logout



Lego
Category: Other
Price: \$151.0
Available Quantity: 100
Posted Date: 2025-04-04 00:03
Description: LEGOs are plastic building-block toys that rose to massive popularity in the mid-20th century.
Seller: Pavel Sitnik

1

Recommendations

Enter custom amount and place the request

NativeSpark

About Us Home My Account Services Cart Logout

Request Your Custom Amount



Enter Desired Amount:

Lego
Category: Other
Price: \$151.0
Available Quantity: 99
Description: LEGOs are plastic building-block toys that rose to massive popularity in the mid-20th century.

To see the requested press My requests in Services dropdown.

NativeSpark

About Us Home My Account Services ▾ Cart Logout

My Custom Requests

Lego

Requested Amount: 100

Seller: Pavel Sitnik (pavel@mail.ru)

Date of Request: 2025-04-04 10:33

Lego

Requested Amount: 1000

Seller: Pavel Sitnik (pavel@mail.ru)

Date of Request: 2025-04-10 22:10

Saved

NativeSpark

About Us Home My Account Services ▾ Cart Logout

Saved Listings

Saved Products

 About Arm

Category: Electronics

Price: \$208.34

Description: Sister purpose six computer challenge. Billion fear spring note quickly still kitchen.

[View Product](#)

 Full Spend

Category: Clothing

Price: \$152.97

Description: Again religious thousand enjoy certainly. Cup person finish test.

[View Product](#)



Lego

Category: Other

Price: \$151.0

Description: LEGOs are plastic building-block toys that rose to massive popularity in the mid-20th century.

[View Product](#)

Subscription
My Orders
My Postings
My Requests
Saved

My Postings

The Business User can post/edit/delete jobs and projects.

The screenshot shows the 'My Postings' section of the NativeSpark platform. At the top, there's a navigation bar with links: About Us, Home, My Account, Services, Cart, and Logout. A dropdown menu is open under 'Services', showing options like Subscription, My Orders, My Postings (which is highlighted with a red box), My Requests, and Saved. Below the navigation, the title 'My Postings' is displayed, with tabs for 'Job Postings' and 'Project Postings'. There are two job posting entries:

- Job Posting 1:** Title: hello. Company: Anyvisa services corp. Location: jsghhnjts. Employment Type: Full-time. Salary: \$34.0. Required Experience: 4. Required Skills: hello. Posted On: 2025-04-06 19:11. Description: hello. Actions: Edit (green button) and Delete (red button).
- Job Posting 2:** Title: bye. Company: Anyvisa services corp. Location: jsghhnjts. Employment Type: Full-time. Salary: \$89.0. Required Experience: 7. Required Skills: bye. Posted On: 2025-04-06 19:12. Description: bye. Actions: Edit (green button) and Delete (red button).

An 'Add Job Posting' button is located at the top right of the posting area.

The screenshot shows the 'My Postings' section of the NativeSpark platform, specifically the 'Project Postings' tab. At the top, there's a navigation bar with links: About Us, Home, My Account, Services, Cart, and Logout. A 'Back to Account Page' link is visible in the top right. Below the navigation, the title 'My Postings' is displayed, with tabs for 'Job Postings' and 'Project Postings' (which is selected). There are two project posting entries:

- Project Posting 1:** Title: hello. Client: Anyvisa services corp. Budget: \$76.0. Deadline: 2025-04-24. Required Skills: hello. Scope: hello. Posted On: 2025-04-06 19:12. Description: hello. Actions: Edit (green button) and Delete (red button).
- Project Posting 2:** Title: bye. Client: Anyvisa services corp. Budget: \$9.0. Deadline: 2025-04-17. Required Skills: bye. Scope: bye. Posted On: 2025-04-06 19:13. Description: bye. Actions: Edit (green button) and Delete (red button).

An 'Add Project Posting' button is located at the top right of the posting area.

Entrepreneur User

Registration

Entrepreneur Registration - Step 2

First Name

Last Name

Select Indigenous Identity

Tell us about yourself

Upload Profile Photo

Choose File No file chosen

Register

My Account

In my account you can edit information about yourself.

NativeSpark

About Us Home My Account Services Cart Logout

My Account

Email e1@gmail.com

User Type ENTREPRENEUR

Subscription Free

First Name Pavel

Last Name Sitrnik

Identity Type First Nations

About Craftsman

Upload New Logo Choose File No file chosen

Edit Profile **Save Changes** **Cancel**

© 2025 NATIVESPARK INTEGRATION SOLUTIONS CORP. All Rights Reserved.

Home Page

Users can save their favourite products, jobs and projects.

Entrepreneurs users on the home page can filter products by existing categories or by keywords searched in the product description and name.

The screenshot shows the NativeSpark Marketplace home page. At the top, there is a navigation bar with links: About Us, Home (which is highlighted with a red box), My Account, Services, Cart, and Logout. Below the navigation bar is a search bar with placeholder text "Search jobs, projects, or products..." and a "Search" button (also highlighted with a red box). A large "Marketplace" heading is centered above the content area. On the left, there is a sidebar with three tabs: "Products" (highlighted with a red box), "Jobs", and "Projects". Below the tabs is a dropdown menu titled "Products" with "All Categories" selected. Under "All Categories", "Electronics" is highlighted with a red box. Other categories listed include Clothing, Books, Sports, Home, and Other. To the right of the dropdown is a "Filter" button. The main content area displays two product cards. The first card is for a "Central Bank" product, which is categorized under Electronics. It includes a small image, a "Product Image" link, a price of \$89.45, and a detailed description: "Defense believe man minute. Difference employee wonder executive outside region after network. Sister purpose six computer challenge. Billion fear spring note quickly still kitchen." The second card is for a "Box Hope" product, categorized under Books. It includes a small image, a "Product Image" link, a price of \$384.45, and a detailed description: "Which shoulder cup information loss. Option actually put stay gas think series. History main south task available at. Important fear start. Peace true establish clearly." Both cards have a "Category" label followed by the category name and a "Price" label followed by the price.

Entrepreneurs users on the home page can filter jobs by existing categories or by keywords searched in the description and name.

The screenshot shows the NativeSpark Marketplace home page. At the top, there is a navigation bar with links: About Us, Home (which is highlighted with a red box), My Account, Services, Cart, and Logout. Below the navigation bar is a search bar with placeholder text "Search jobs, projects, or products..." and a "Search" button (also highlighted with a red box). A large "Marketplace" heading is centered above the content area. On the left, there is a sidebar with three tabs: "Products", "Jobs" (highlighted with a red box), and "Projects". Below the tabs is a dropdown menu titled "Job Postings" with "All Job Types" selected. Under "All Job Types", "Full-time" is highlighted with a red box. Other options listed include Part-time, Internship, and a location field containing "Location: jsghnnjts". To the right of the dropdown is a "Filter" button. The main content area displays two job posting cards. The first card is for a job listing with a "Location" of "jsghnnjts", an "Employment Type" of "Full-time", a "Salary" of "\$34.0", and "Skills Required" of "hello". The second card is for another job listing with a "Location" of "jsghnhnjts", an "Employment Type" of "Full-time", a "Salary" of "\$89.0", and "Skills Required" of "bye". Both cards have a "Category" label followed by the location and employment type, and a "Price" label followed by the salary.

Entrepreneurs users on the home page can filter projects by existing categories or by keywords searched in the description and name.

NativeSpark

About Us Home My Account Services ▾ Cart Logout

Marketplace

Products Jobs Projects

Search jobs, projects, or products... **Search**

Project Postings

 hello

Budget: \$76.0

Deadline: 2025-04-24

Skills Required: hello

 bye

Budget: \$9.0

Deadline: 2025-04-17

Skills Required: bye

Additionally, by clicking on the product image or name, a detailed summary will open.

NativeSpark

About Us Home My Account Services ▾ Cart Logout



Lego
Category: Other
Price: \$151.0
Available Quantity: 100
Posted Date: 2025-04-04 00:03
Description: LEGOs are plastic building-block toys that rose to massive popularity in the mid-20th century.
Seller: Pavel Sitnik
 Add to Cart
Request Custom Amount Back to Home

Recommendations

hello

Location: jsghhnjts

Employment Type: Full-time

Salary: \$34.0

Required Experience: 4

Skills Required: hello

Description: hello

Posted By: Anyvisa services corp.

[Back to Home](#)

© 2025 NATIVESPARK INTEGRATION SOLUTIONS CORP. All Rights Reserved.

hello

Budget: \$76.0

Deadline: 2025-04-24

Skills Required: hello

Project Scope: hello

Description: hello

Posted By: Anyvisa services corp.

[Back to Home](#)

© 2025 NATIVESPARK INTEGRATION SOLUTIONS CORP. All Rights Reserved.

Cart

The product can be added to cart and proceeded to check out.

Your Cart

Product Name	Description	Price	Quantity	Total	Actions
Lego	LEGOs are plastic building-block toys that rose to massive popularity in the mid-20th century.	151.0	<input type="button" value="1"/>	151.0	Remove

[Proceed to Checkout](#)

On summary page the information about seller and the product will be shown.

The screenshot shows a summary page for a product purchase. At the top, there's a navigation bar with links: About Us, Home, My Account, Services, Cart, and Logout. Below the navigation, there are three main sections: Seller Info, Product Info, and Summary. The Seller Info section contains the name and email of the seller. The Product Info section lists the item as 'Lego', price '\$151.0', quantity '1', and total '\$151.00'. The Summary section displays the total items '1' and total price '\$151.0'. A large red box highlights the 'Proceed to Checkout' button in the Summary section.

To place the order shipping information needed.

The screenshot shows a checkout page. At the top, there's a navigation bar with links: About Us, Home, My Account, Services, Cart, and Logout. Below the navigation, there are two main sections: Shipping Information and Payment Details. The Shipping Information section includes fields for Country, City, Province, Street Address, Postal Code, and Phone Number. The Payment Details section includes fields for Card Number, Expiration Date (MM/YY), CVV, and a large 'Confirm Purchase' button. A dark green footer bar at the bottom contains the copyright notice: © 2025 NATIVESPARK INTEGRATION SOLUTIONS CORP. All Rights Reserved.

Services dropdown

The screenshot shows the NativeSpark Marketplace website. At the top, there is a dark green header bar with the "NativeSpark" logo on the left and navigation links for "About Us", "Home", "My Account", "Services", "Cart", and "Logout" on the right. A red box highlights the "Services" dropdown menu, which is currently open and displays the following options: "Subscription", "My Orders", "Buyers of My Products", "Requests", "My Postings", "My Requests", and "Saved". Below the header, the main content area is titled "Marketplace" and features tabs for "Products", "Jobs", and "Projects". A search bar with placeholder text "Search jobs, projects, or products..." is also present. Under the "Products" section, there is a dropdown menu for "All Categories" and a "Filter" button.

My Orders

To see the information about purchased products click My Orders

The screenshot shows the "My Orders" page on the NativeSpark Marketplace. The page title is "My Orders". The content is organized into three separate sections, each representing a purchase. The first purchase is for "Lego" items:

Full Spend
Quantity: 2
Total Price: \$305.94
Seller: hillstephen@example.com
Phone Number: 45678
Status: SUCCESS
Shipping Address: 216 ltbleu, hhighfgzg, bc, canada, 5tghb7
Date: 2025-04-04 11:18

The second purchase is for "Lego" items:

Lego
Quantity: 1
Total Price: \$151.00
Seller: pavel@mail.ru
Phone Number: 45678
Status: SUCCESS
Shipping Address: 216 ltbleu, hhighfgzg, bc, canada, 5tghb7
Date: 2025-04-04 11:18

The third purchase is for "Lego" items:

Lego
Quantity: 2
Total Price: \$302.00
Seller: pavel@mail.ru
Phone Number: 45678
Status: SUCCESS
Shipping Address: 216 ltbleu, van, bc, canada, 5tghb7
Date: 2025-04-04 15:04

My Requests

If the product is out of stock the user can request custom amount.

The screenshot shows a product page for 'About Arm'. The product details are as follows:

- Category:** Electronics
- Price:** \$208.34
- Available Quantity:** 0
- Posted Date:** 2024-06-30 19:07

Below the details, there is a 'Product Image' placeholder and a description: "Sister purpose six computer challenge. Billion fear spring note quickly still kitchen.". The seller information shows "Seller: Andrew Kent" and "Coming Soon...". A yellow button labeled "Request Custom Amount" is present, and a dark green button labeled "Back to Home" is at the bottom.

Enter custom amount and place the request

The screenshot shows a form titled "Request Your Custom Amount" for the same product. The product details are identical to the previous page. The form includes a "Product Image" placeholder, an input field for "Enter Desired Amount" containing the value "10", and a yellow "Submit Request" button. A dark green "Back to Home" button is also visible.

To see the requested press My requests in Services dropdown.

The screenshot shows the "My Custom Requests" section. It displays a single request for the 'About Arm' product. The request details are:

- About Arm**
- Requested Amount:** 1
- Seller:** Andrew Kent (oalvarado@example.net)
- Date of Request:** 2025-04-10 22:30

Saved

Browse saved products/jobs/products

The screenshot shows the NativeSpark application interface. At the top, there is a dark green header bar with the 'NativeSpark' logo on the left and navigation links for 'About Us', 'Home', 'My Account', 'Services', 'Cart', and 'Logout' on the right. A dropdown menu is open on the right side of the header, listing options: 'Subscription', 'My Orders', 'Buyers of My Products', 'Requests', 'My Postings', 'My Requests', and 'Saved'. The 'Saved' option is highlighted with a red box. Below the header, the main content area has a title 'Saved Listings' and three tabs: 'Products' (selected), 'Jobs', and 'Projects'. Underneath, a section titled 'Saved Products' displays a single product card. The card includes a placeholder image labeled 'Product Image', the brand name 'About Arm', category 'Electronics', price '\$208.34', and a description: 'Sister purpose six computer challenge. Billion fear spring note quickly still kitchen.'. A 'View Product' button is at the bottom of the card.

My Postings

The Entrepreneurs User can post/edit/delete products.

The screenshot shows the NativeSpark application interface. At the top, there is a dark green header bar with the 'NativeSpark' logo on the left and navigation links for 'About Us', 'Home', 'My Account', 'Services', 'Cart', and 'Logout' on the right. A dropdown menu is open on the right side of the header, listing options: 'Subscription', 'My Orders', 'Buyers of My Products', 'Requests', 'My Postings' (selected), 'My Requests', and 'Saved'. The 'My Postings' option is highlighted with a red box. Below the header, the main content area has a title 'My Postings' and a section titled 'Products for Sale'. It displays a product card for a 'Lego' item. The card includes a placeholder image labeled 'Product Image', the brand name 'Lego', category 'Other', price '\$100.0', QTY: 100, and a timestamp 'Posted On: 2025-04-10 22:40'. A detailed description follows: 'Description: LEGOs are plastic building-block toys that rose to massive popularity in the mid-20th century.' To the right of the card, there are two buttons: 'Edit' and 'Delete', with 'Edit' being green and 'Delete' being red.

Buyers of my products

The user can see the buyers of the product for future shipping.

The screenshot shows the NativeSpark application interface. At the top, there is a dark green header bar with the 'NativeSpark' logo on the left and navigation links on the right: 'About Us', 'Home', 'My Account', 'Services ▾', 'Cart', and 'Logout'. A dropdown menu is open from the 'Services' link, showing options like 'Subscription', 'My Orders', 'Buyers of My Products' (which is highlighted with a red box), 'Requests', 'My Postings', 'My Requests', and 'Saved'. Below the header, the main content area has a title 'People Who Bought My Products'. Underneath, there is a card containing product details: 'Lego', 'Quantity: 1', 'Price: \$100.0', 'Total Price: \$100.0', 'Buyer Email: val@mail.ru', 'Phone: 6045006272', 'Shipping Address: 216 1tbleu, van, bc, canada, 5tghb7', and 'Date: 2025-04-10 22:41'.

Requests

The user can see the requests of the product for future manufacturing.

The screenshot shows the NativeSpark application interface. At the top, there is a dark green header bar with the 'NativeSpark' logo on the left and navigation links on the right: 'About Us', 'Home', 'My Account', 'Services ▾', 'Cart', and 'Logout'. A dropdown menu is open from the 'Services' link, showing options like 'Subscription', 'My Orders', 'Buyers of My Products', 'Requests' (which is highlighted with a red box), 'My Postings', 'My Requests', and 'Saved'. Below the header, the main content area has a title 'Custom Requests for My Products'. Underneath, there is a card containing request details: 'Lego', 'Requested Amount: 100', 'Requester: val@mail.ru', and 'Requested At: 2025-04-10 22:41'.

Basic User

Registration

Basic User Registration - Step 2

First Name

Last Name

Tell us about yourself

Upload Profile Photo

No file chosen

My Account

In my account you can edit information about yourself.

NativeSpark

About Us Home My Account Services Cart Logout

My Account

Email
bu1@gmail.com

User Type
BASIC

Subscription
Free

First Name
Lera

Last Name
Nikitina

About
Craftsman

Upload New Logo
 No file chosen

© 2025 NATIVESPARK INTEGRATION SOLUTIONS CORP All Rights Reserved.

Home Page

Basic users on the home page can filter products by existing categories or by keywords searched in the product description and name.

The screenshot shows the NativeSpark Marketplace home page. At the top, there is a navigation bar with links: About Us, Home (which is highlighted with a red box), My Account, Services, Cart, and Logout. Below the navigation bar is a search bar labeled "Search jobs, projects, or products..." with a "Search" button. A large red box highlights the search bar. To the right of the search bar is a dropdown menu titled "All Categories" which is also highlighted with a red box. The menu lists categories: All Categories, Electronics, Clothing, Books, Sports, and Home. Below the dropdown is a "Filter" button. The main content area is titled "Marketplace". It displays two product cards. The first product card is for "Central Break" (Category: Clothing, Price: \$89.45). The second product card is for "Box Hope" (Category: Books, Price: \$384.45). Both cards show a small thumbnail image, a heart icon for favoriting, and a "Product Image" link.

Users can save their favourite products for later purchase or set a custom amount if their desired quantity exceeds the stock available at that moment.

This screenshot shows the same NativeSpark Marketplace interface as the previous one, but with different product cards. The first product card for "About Arm" (Category: Electronics, Price: \$208.34) has its favorite icon (a red heart inside a square) highlighted with a red box. The second product card for "Central Break" (Category: Clothing, Price: \$89.45) also has its favorite icon highlighted with a red box. The rest of the interface, including the navigation bar, search bar, and other product details, remains consistent with the first screenshot.

Additionally, by clicking on the product image or name, a detailed summary will open.



The screenshot shows a product detail page for 'Lego'. At the top, there's a large image of the LEGO logo. Below it, the word 'Lego' is displayed in a bold, black font. To the right of the image, there are several details: 'Category: Other', 'Price: \$151.0', 'Available Quantity: 100', 'Posted Date: 2025-04-04 00:03', and a description stating 'LEGOs are plastic building-block toys that rose to massive popularity in the mid-20th century.' Below these details are buttons for 'Add to Cart' (with a quantity input field set to 1), 'Request Custom Amount', and 'Back to Home'.

Recommendations

Cart

The product can be added to cart and proceeded to check out.



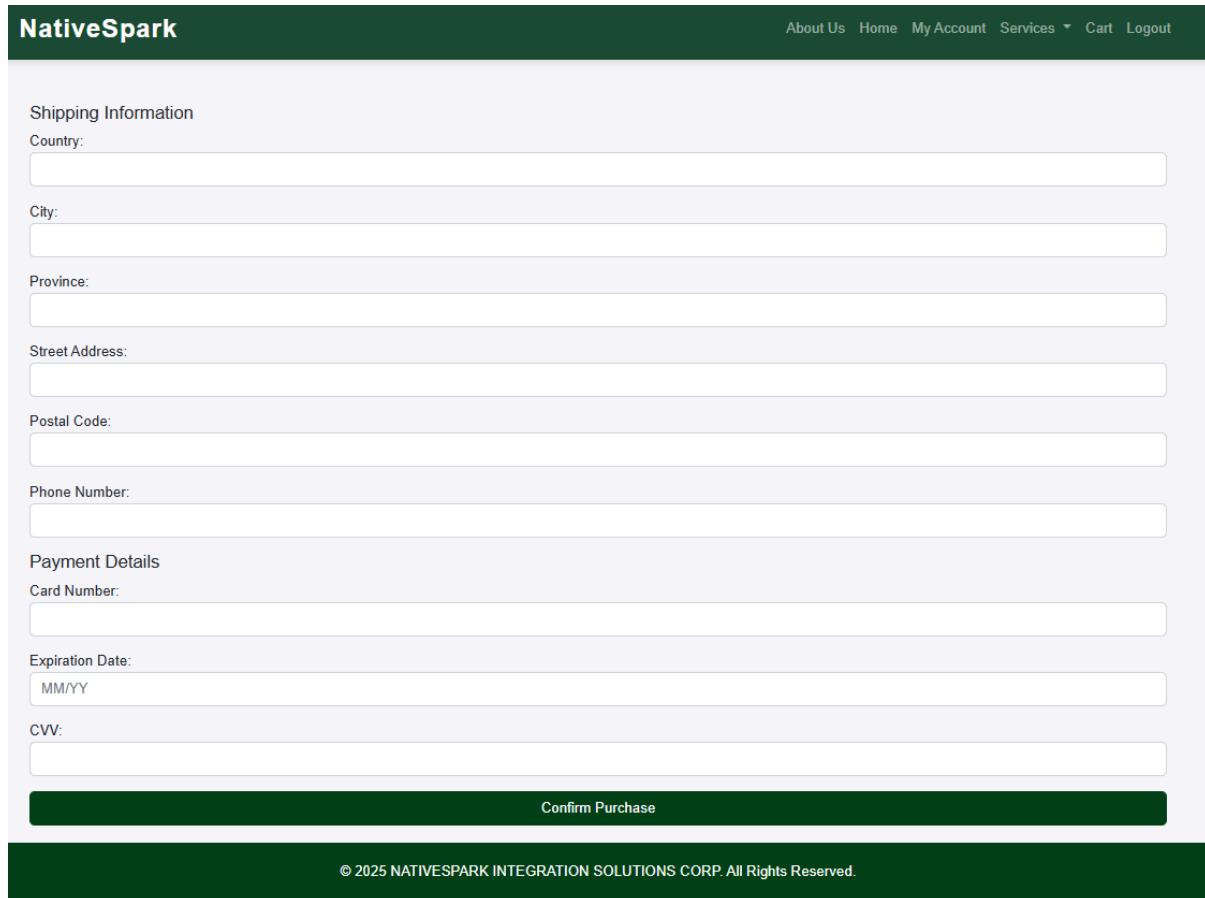
The screenshot shows a 'Your Cart' page. It features a table with columns for Product Name, Description, Price, Quantity, Total, and Actions. There is one item in the cart: 'Lego' with a description of 'LEGOs are plastic building-block toys that rose to massive popularity in the mid-20th century.' The price is \$151.0, the quantity is 1, and the total is \$151.0. Action buttons include 'Update' and 'Remove'. Below the table is a 'Proceed to Checkout' button.

On summary page the information about seller and the product will be shown.



The screenshot shows a summary page with three main sections: 'Seller Info', 'Product Info', and 'Summary'. The 'Seller Info' section contains the seller's name (Pavel Sitnik) and email (pavel@mail.ru). The 'Product Info' section contains the product name (Lego), price (\$151.0), quantity (1), and total (\$151.00). The 'Summary' section contains a 'Summary' heading, 'Total Items: 1', 'Total Price: \$151.0', and a 'Proceed to Checkout' button, which is highlighted with a red box.

To place the order shipping information needed.



The screenshot shows a "Shipping Information" form on the NativeSpark website. It includes fields for Country, City, Province, Street Address, Postal Code, and Phone Number. Below this is a "Payment Details" section with fields for Card Number, Expiration Date (MM/YY), and CVV. A "Confirm Purchase" button is at the bottom. The entire page has a dark green header and footer.

Shipping Information

Country:

City:

Province:

Street Address:

Postal Code:

Phone Number:

Payment Details

Card Number:

Expiration Date:

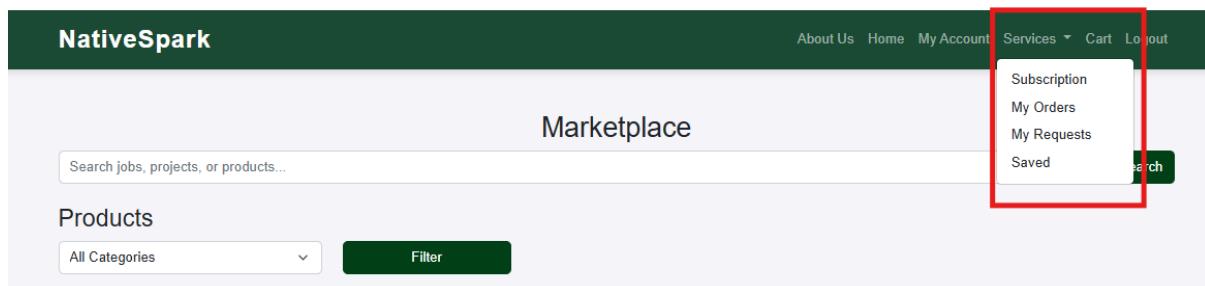
MM/YY

CVV:

Confirm Purchase

© 2025 NATIVESPARK INTEGRATION SOLUTIONS CORP All Rights Reserved.

Services dropdown



The screenshot shows the NativeSpark Marketplace. A red box highlights the "Services" dropdown menu in the top right corner, which contains links for Subscription, My Orders, My Requests, and Saved. The main page features a search bar, a "Products" section with a dropdown menu and a "Filter" button, and a "Marketplace" heading.

NativeSpark

About Us Home My Account Services Cart Logout

Marketplace

Search jobs, projects, or products...

Products

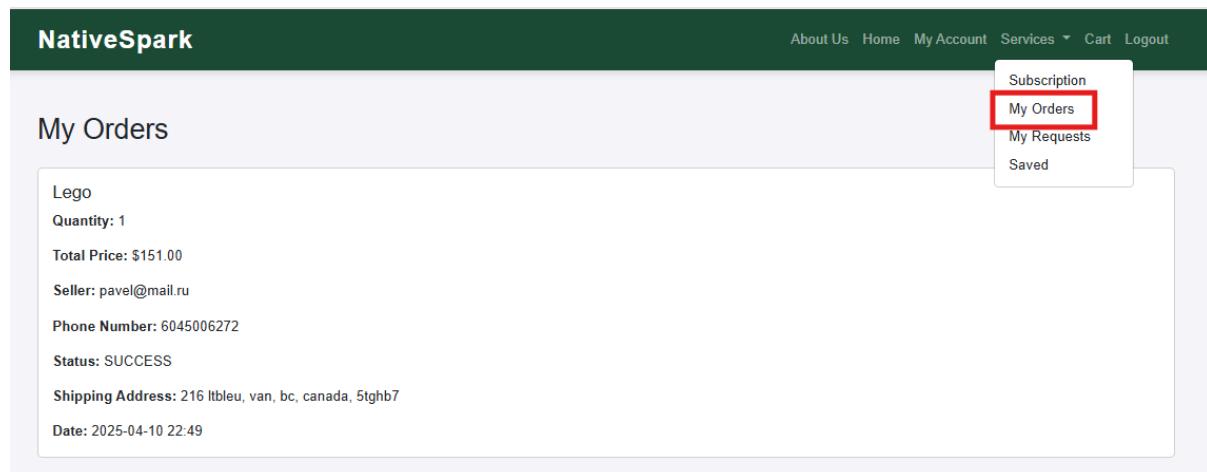
All Categories Filter

Services

Subscription
My Orders
My Requests
Saved

My Orders

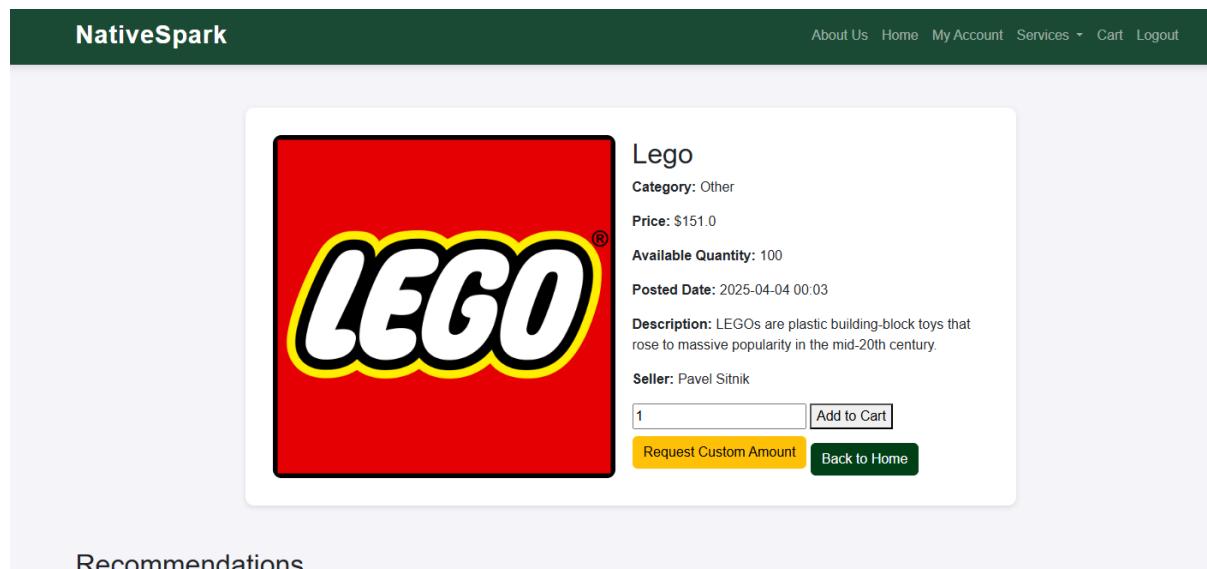
To see the information about purchased products click My Orders



The screenshot shows the 'NativeSpark' mobile application interface. At the top, there's a dark green header bar with the 'NativeSpark' logo on the left and navigation links for 'About Us', 'Home', 'My Account', 'Services', 'Cart', and 'Logout' on the right. Below the header, a white content area has a title 'My Orders'. Inside this area, there's a product listing for a 'Lego' item. The listing includes: 'Quantity: 1', 'Total Price: \$151.00', 'Seller: pavel@mail.ru', 'Phone Number: 6045006272', 'Status: SUCCESS', 'Shipping Address: 216 1tbleu, van, bc, canada, 5tghb7', and 'Date: 2025-04-10 22:49'. To the right of the product listing, a context menu is displayed with options: 'Subscription', 'My Orders' (which is highlighted with a red border), 'My Requests', and 'Saved'. The 'My Orders' option is the one currently selected.

My Requests

To request custom amount the user needs to submit it on product details page.



The screenshot shows the 'NativeSpark' mobile application interface. At the top, there's a dark green header bar with the 'NativeSpark' logo on the left and navigation links for 'About Us', 'Home', 'My Account', 'Services', 'Cart', and 'Logout' on the right. Below the header, a white content area displays a product detail page for a 'Lego' item. The product image is a red square with the 'LEGO' logo. To the right of the image, the product name is 'Lego', and its details are listed: 'Category: Other', 'Price: \$151.0', 'Available Quantity: 100', 'Posted Date: 2025-04-04 00:03', and 'Description: LEGOs are plastic building-block toys that rose to massive popularity in the mid-20th century.' Below these details are buttons for 'Seller: Pavel Sitrnik', a quantity input field containing '1', an 'Add to Cart' button, a yellow 'Request Custom Amount' button (which is highlighted with a red border), and a 'Back to Home' button.

Recommendations

Enter custom amount and place the request

NativeSpark

About Us Home My Account Services ▾ Cart Logout

Request Your Custom Amount



Enter Desired Amount:
1000

Submit Request Back to Home

Lego
Category: Other
Price: \$151.0
Available Quantity: 99
Description: LEGOs are plastic building-block toys that rose to massive popularity in the mid-20th century.

To see the requested press My requests in Services dropdown.

NativeSpark

About Us Home My Account Services ▾ Cart Logout

My Custom Requests

Lego
Requested Amount: 100
Seller: Pavel Sitnik (pavel@mail.ru)
Date of Request: 2025-04-04 10:33

Lego
Requested Amount: 1000
Seller: Pavel Sitnik (pavel@mail.ru)
Date of Request: 2025-04-10 22:10

Saved

NativeSpark

About Us Home My Account Services Cart Logout

Saved Listings

Saved Products

 About Arm
Category: Electronics
Price: \$208.34
Description: Sister purpose six computer challenge. Billion fear spring note quickly still kitchen.
[View Product](#)

 Full Spend
Category: Clothing
Price: \$152.97
Description: Again religious thousand enjoy certainly. Cup person finish test.
[View Product](#)

 Lego
Category: Other
Price: \$151.0
Description: LEGOs are plastic building-block toys that rose to massive popularity in the mid-20th century.
[View Product](#)

Subscription
My Orders
My Postings
My Requests
Saved