

CSIS 4495 - Applied Research Project

Midterm Report

**NATIVESPARK INTEGRATION SOLUTIONS CORP.**



Valeriia Nikitina (300374609)

Video Link: <https://youtu.be/i0InDv3HUi0>

## Contents

Introduction.....	3
The origin of the idea.....	3
Problem framing .....	3
Competitive advantage among others .....	4
Overview of the platform.....	5
The architecture of the system .....	6
The hardware and software configuration of the system .....	6
Hardware Requirements.....	6
Software Configuration.....	6
Changes made to the original proposal .....	8
Project Planning and Timeline .....	9
Development Phase.....	9
Testing Phase.....	9
Final Submission and Presentation .....	9
Gantt Chart.....	10
Implemented Features .....	11
User Registration for All Three User Types.....	11
Step 1: Basic Registration Form .....	13
Step 2: Role-Specific Registration.....	14
Password Encryption .....	17
Existing Database Schema .....	17
Login Handling .....	19
Session-Based Authentication.....	20
Welcome page .....	21
Dynamic Navbar .....	22
Hours logs .....	23
References.....	28

# 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.

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.
- **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.
- OpenAI: <https://github.com/openai/openai-java>, <https://platform.openai.com/docs/>
- Spring Boot WebClient: <https://spring.io/guides/gs/reactive-rest-service/>
- **Smile Java Library**: <https://haifengl.github.io/>
- **Bootstrap Chat Widgets**: <https://getbootstrap.com/docs/5.3/examples/chat/>

## 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

During development, no significant changes were made to the database structure; instead, three additional tables were introduced to avoid consolidating everything into a single users table. This adjustment was essential for maintaining database normalization, specifically Third Normal Form (3NF), ensuring that each user type (BasicUser, BusinessUser, and EntrepreneurUser) had its own table with relevant attributes. The Table per Subclass strategy was employed to eliminate redundant data and excessive null values while retaining the core user information in the users table. This approach enhanced data integrity, scalability, and maintainability without modifying the core logic of the program.



# Project Planning and Timeline

## 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:** Functional implementation of all platform features (e.g., AI-matchmaking, chatbot, admin dashboard if time permits).

## Testing Phase

**Milestone:** Refinement and testing (April 6)

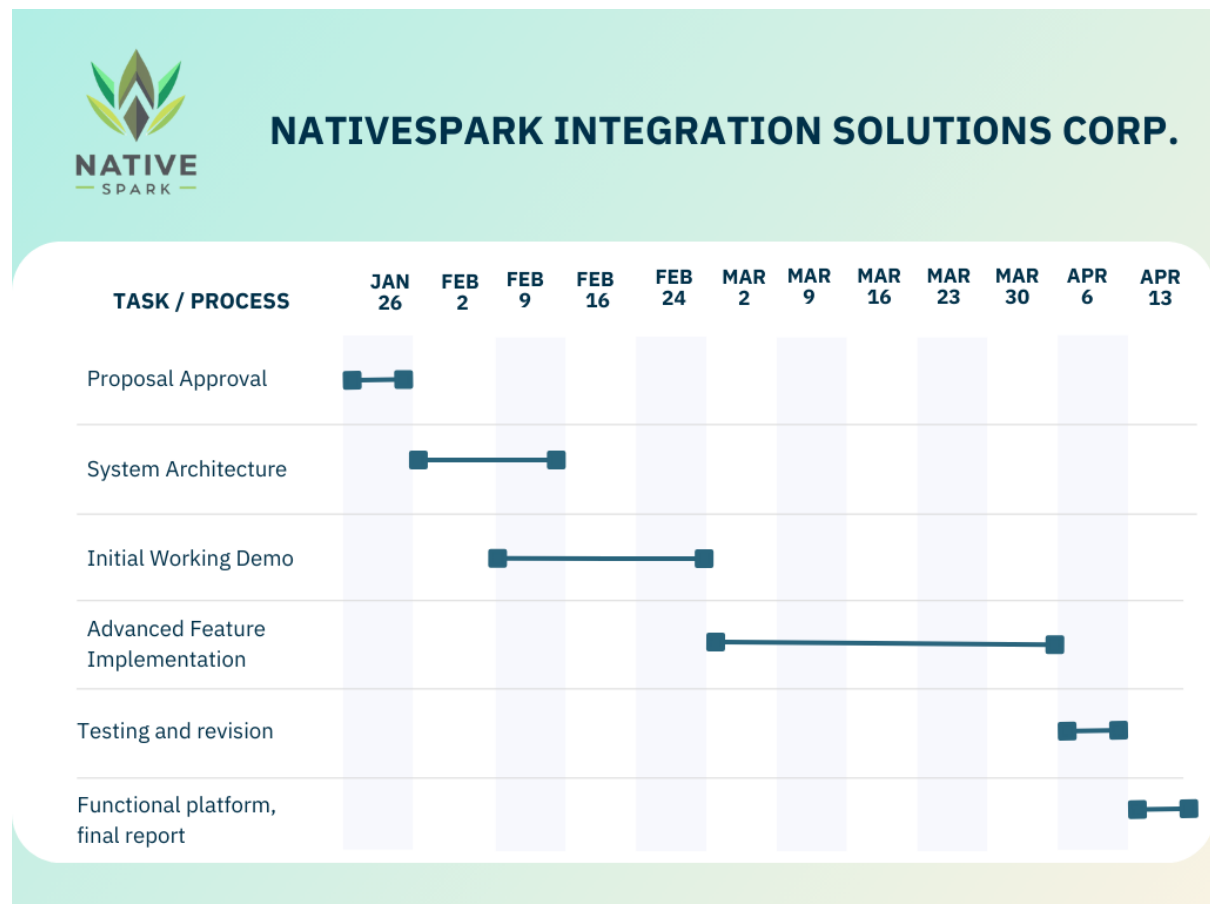
**Deliverables:** Usability testing and revision.

## Final Submission and Presentation

**Milestone:** Final Submission and Presentation (Apr 13)

**Deliverables:** Fully functional platform, final report.

# Gantt Chart



# Implemented Features

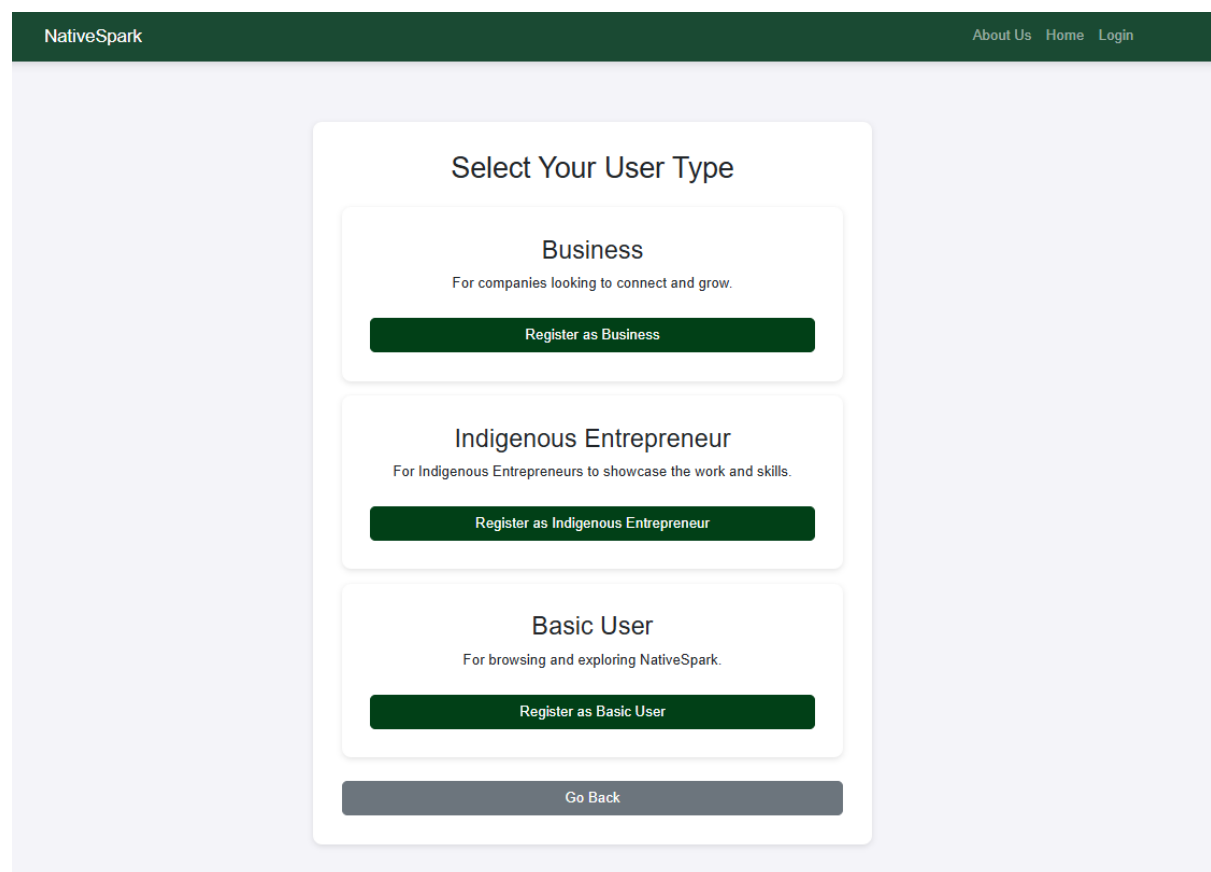
## User Registration for All Three User Types

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.



The screenshot displays the 'Select Your User Type' registration page for NativeSpark. The page features a dark green header with the 'NativeSpark' logo on the left and navigation links 'About Us', 'Home', and 'Login' on the right. The main content area is a light purple gradient. In the center, there is a white card titled 'Select Your User Type'. This card contains three distinct registration options, each with a title, a descriptive subtitle, and a dark green 'Register as' button. The options are: 'Business' (For companies looking to connect and grow.), 'Indigenous Entrepreneur' (For Indigenous Entrepreneurs to showcase the work and skills.), and 'Basic User' (For browsing and exploring NativeSpark.). At the bottom of the card is a grey 'Go Back' button.

User Type	Description	Registration Button
Business	For companies looking to connect and grow.	Register as Business
Indigenous Entrepreneur	For Indigenous Entrepreneurs to showcase the work and skills.	Register as Indigenous Entrepreneur
Basic User	For browsing and exploring NativeSpark.	Register as Basic User

```

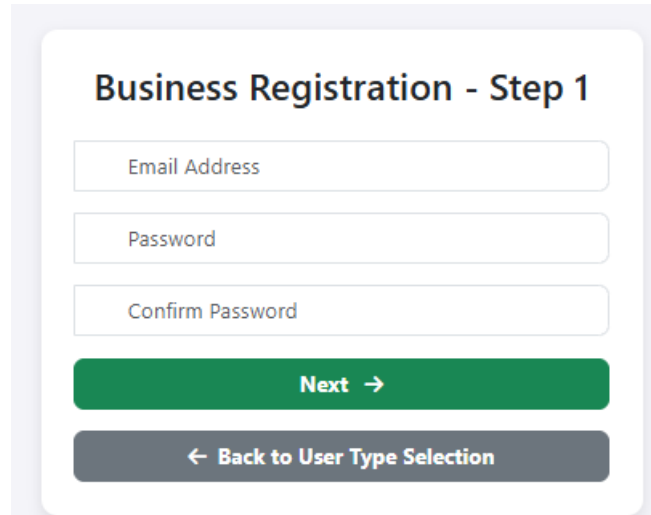
6 public class UserRegistrationController {
7     @PostMapping("/register/user-type")  valeriiianik
8     public String selectUserType(@RequestParam("userType") String userType, HttpSession session) {
9         session.setAttribute("selectedUserType", userType);
10
11         if ("BUSINESS".equals(userType)) {
12             return "redirect:/register/business";
13         }
14         else if ("ENTREPRENEUR".equals(userType)) {
15             return "redirect:/register/entrepreneur";
16         } else if ("BASIC".equals(userType)){
17             return "redirect:/register/basic";
18         }
19
20         return "redirect:/register/basic";
21     }
22
23     @GetMapping("/register/business")  valeriiianik
24     public String showBusinessRegistrationForm() {
25         return "business-register"; //
26     }
27
28     @GetMapping("/register/entrepreneur")  valeriiianik
29     public String showEntrepreneurRegistrationForm() {
30         return "entrepreneur-register"; //
31     }
32
33     @GetMapping("/register/basic")  valeriiianik
34     public String showBasicRegistrationForm() {
35         return "basic-register"; //
36     }
37 }

```

## 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 web form titled "Business Registration - Step 1". It contains three input fields: "Email Address", "Password", and "Confirm Password". Below the fields are two buttons: a green "Next →" button and a grey "← Back to User Type Selection" button.

```
@PostMapping("/step1")  @ valeriiianik *
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";
    }

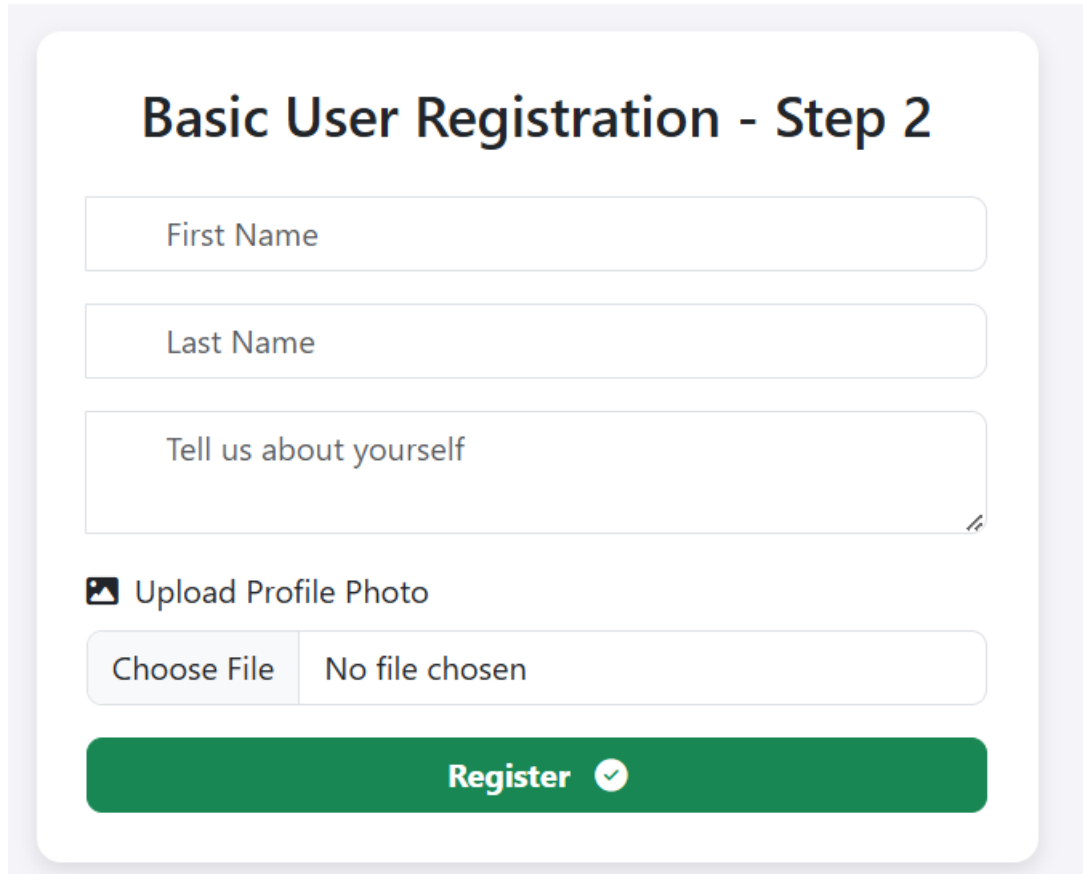
    User user = userService.createUser(email, password, userType: "BUSINESS");
    session.setAttribute("registeredUser", user);

    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.



The form is titled "Basic User Registration - Step 2". It contains three input fields: "First Name", "Last Name", and "Tell us about yourself". Below these is a section for "Upload Profile Photo" with a "Choose File" button and a "No file chosen" text. At the bottom is a green "Register" button with a checkmark icon.

```
@PostMapping("/step2")  valeriiianik
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(s: "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(s: "registeredUser");
    return "redirect:/login?success=basic_registered";
}
```


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

## Business Registration - Step 2

 Upload Logo

Choose File

No file chosen

Register 

```
@PostMapping("/{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

 Upload Profile Photo

Choose File

No file chosen

**Register** 

```

@PostMapping("/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

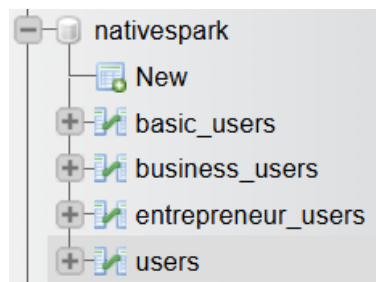
\$2a\$10\$pOqLlmvI470Aob.IZ/0jvufhv5CTAaxDZczG.RFU8kd...

\$2a\$10\$8a5QTTRffYtw5lgikl6wvOVxN06FeOmuUzK5u5tKCKQ...

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

## Existing Database Schema

The platform follows database normalization (3NF), ensuring that user data is efficiently stored without redundancy. Instead of a single user table, there are three user-specific tables:



- **users** (Stores basic user credentials)

user_id	email	password	user_type
1	lera@gmail.com	\$2a\$10\$pOqLlmvI470Aob.IZ/0jvufhv5CTAaxDZczG.RFU8kd...	BASIC
2	val@mail.ru	\$2a\$10\$8a5QTTRffYtw5lgikl6wvOVxN06FeOmuUzK5u5tKCKQ...	BUSINESS
3	pavel@mail.ru	\$2a\$10\$earbZ2goQnN1SAQamLqg6u8OCIRA4D6vgfUz2LfSg7h...	ENTREPRENEUR

- **basic\_users** (Stores profile information for general users)

id	about	first_name	last_name	photo_path	user_id
1	hgf	Lera	Nikitina	uploads\photos\1740372837749_LEGO_logo.svg.png	1

- **business\_users** (Stores business details)

id	business_name	description	logo_path	user_id
1	Anyvisa services corp.	k,jmhngbfv	uploads\logos\1740372895112_LEGO_logo.svg.png	2

- **entrepreneur\_users** (Stores entrepreneur-specific data)

id	about	first_name	identity_type	last_name	photo_path	user_id
1	mjmhjngf	Pavel	First Nations	Sitnik	uploads\photos\1740372971075_LEGO_logo.svg.png	3

# Login Handling

**NativeSpark**[About Us](#)[Home](#)[Login](#)

## Login

Email Address

Password

Login

[Don't have an account? Sign up here](#)

```
@PostMapping("/login") new *
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 new *
public String showAccountPage(Model model) {
    org.springframework.security.core.Authentication authentication =
        SecurityContextHolder.getContext().getAuthentication();

    if (authentication == null || !authentication.isAuthenticated()) {
        System.out.println("✗ 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("✗ 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";
}
```

**NativeSpark**

About UsHomeMy AccountLogout

Welcome, val@mail.ru  
Your user type: BUSINESS

© 2025 NATIVESPARK INTEGRATION SOLUTIONS CORP. All Rights Reserved.

**NativeSpark**

About UsHomeMy AccountLogout

Welcome, pavel@mail.ru  
Your user type: ENTREPRENEUR

© 2025 NATIVESPARK INTEGRATION SOLUTIONS CORP. All Rights Reserved.

**NativeSpark**

About UsHomeMy AccountLogout

Welcome, lera@gmail.com  
Your user type: BASIC

© 2025 NATIVESPARK INTEGRATION SOLUTIONS CORP. All Rights Reserved.

## Welcome page



### 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.

```
<div class="collapse navbar-collapse" id="navbarSupportedContent">
  <ul class="navbar-nav ms-auto mb-2 mb-lg-0">
    <li class="nav-item">
      <a class="nav-link" th:href="@{/about}">About Us</a>
    </li>
    <li class="nav-item">
      <a class="nav-link" th:href="@{/}">Home</a>
    </li>
    <li class="nav-item" sec:authorize="isAuthenticated()">
      <a class="nav-link" th:href="@{/account}">My Account</a>
    </li>
    <li class="nav-item" sec:authorize="!isAuthenticated()">
      <a class="nav-link" th:href="@{/login}">Login</a>
    </li>
    <li class="nav-item" sec:authorize="isAuthenticated()">
      <a class="nav-link" th:href="@{/logout}">Logout</a>
    </li>
  </ul>
</div>
```

## 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 settings. 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	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.
07.02.2025	2.5	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. 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. 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.



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 <code>Files.copy()</code>, 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>
11.02.2025	2.5	<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.</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 "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 <code>loadUserByUsername(String username)</code> method. This omission caused an error during the authentication setup in SecurityConfig.java.</p> <p>To resolve this, I implemented the missing <code>loadUserByUsername</code> 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> <p>Some modifications in htmls files.</p>
12.02.2025	1	<p>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 <code>event.preventDefault();</code> call in the form submission, which was preventing the form from sending data to the backend. To resolve this issue, I removed the <code>onsubmit</code> event. Once I verified that the business registration flow was working correctly, I 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	<p>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.</p>
14.02.2025	2	<p>Setting up the logging for users. Need to start again.</p>

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 programm logges in via users table. 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 (org.apache.tomcat.util.net.openssl.ciphers.Authentication), ensured I was using org.springframework.security.core.Authentication, 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.

## 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/>