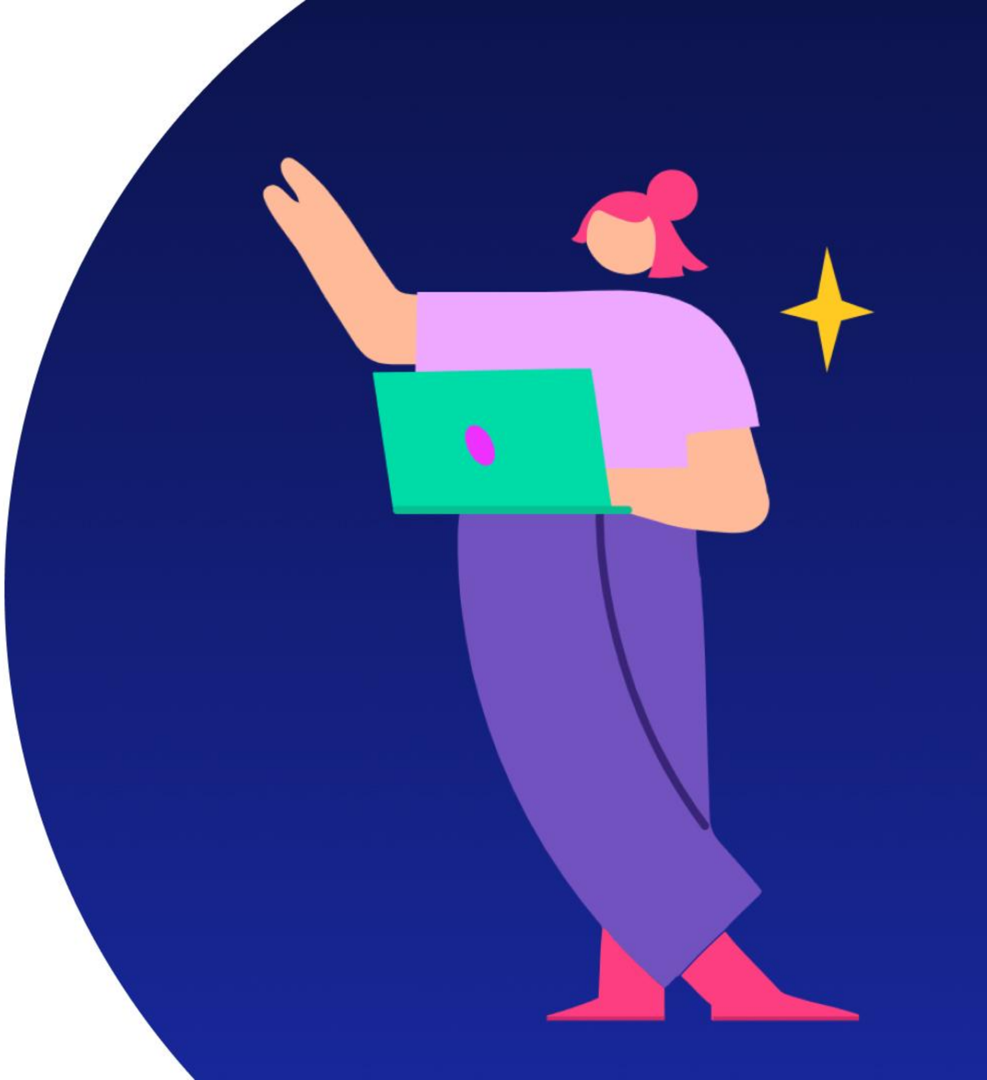
Team 1 presents

JobScout

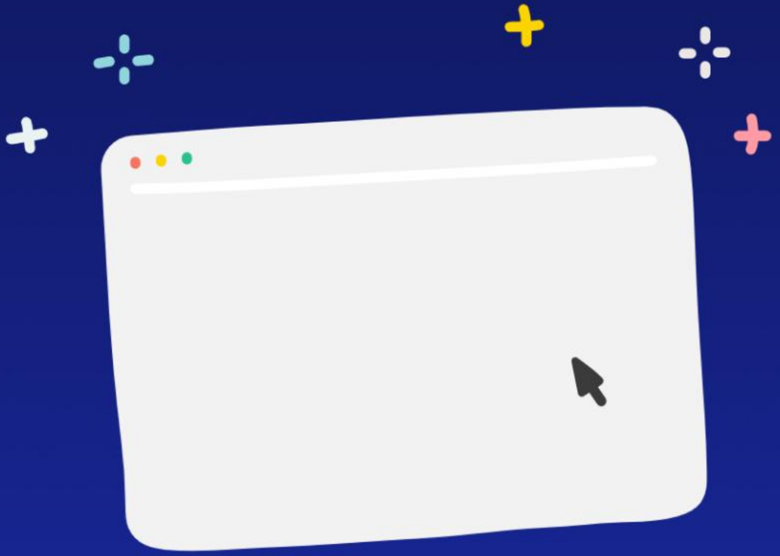
High Fidelity Prototype



Made with passion by:

Ivan, Pavel, Saed and **Viet**

Table of Contents



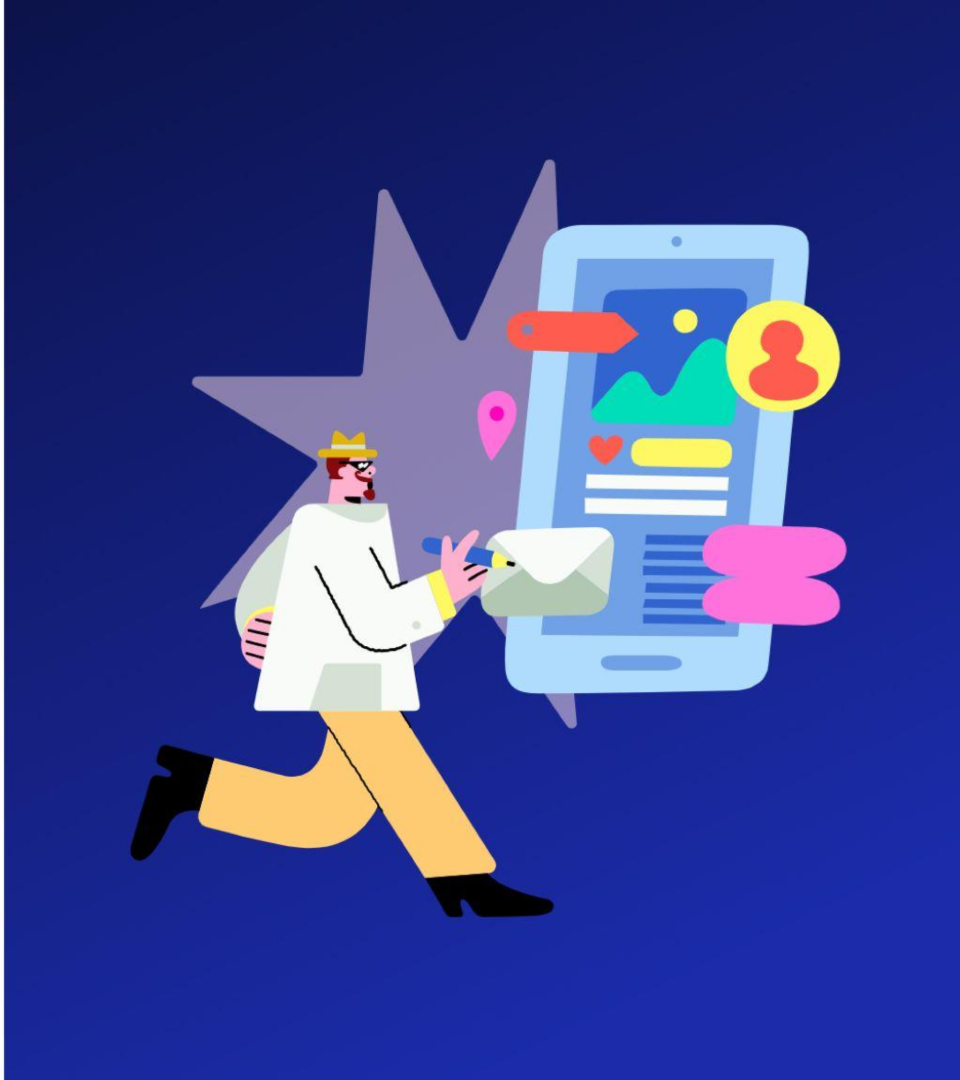
1. Introduction
2. Original Prototype
3. Backend Demo
4. Frontend Demo
5. Sprint Ceremony insights
6. Questions

Introduction

Unified job search at your fingertips: Our high-fidelity prototype streamlines job hunting by scraping 6 platforms into one intuitive app.

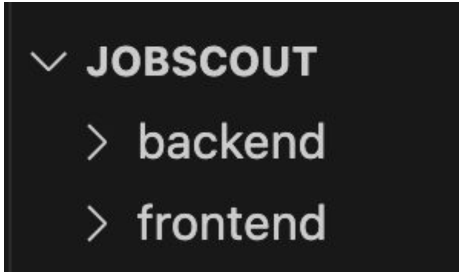
The Devs

Pavel: Frontend
Ivan: Frontend
Viet: Backend
Saed: Backend

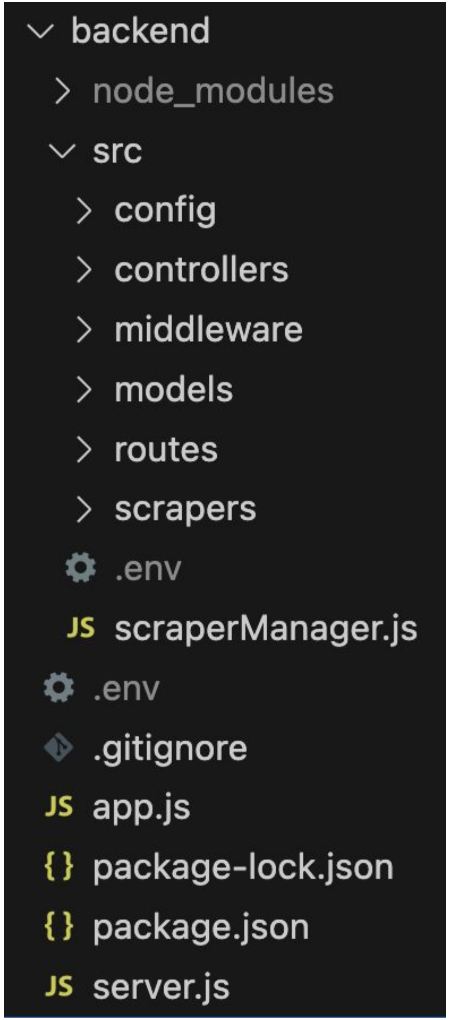


Original Prototype

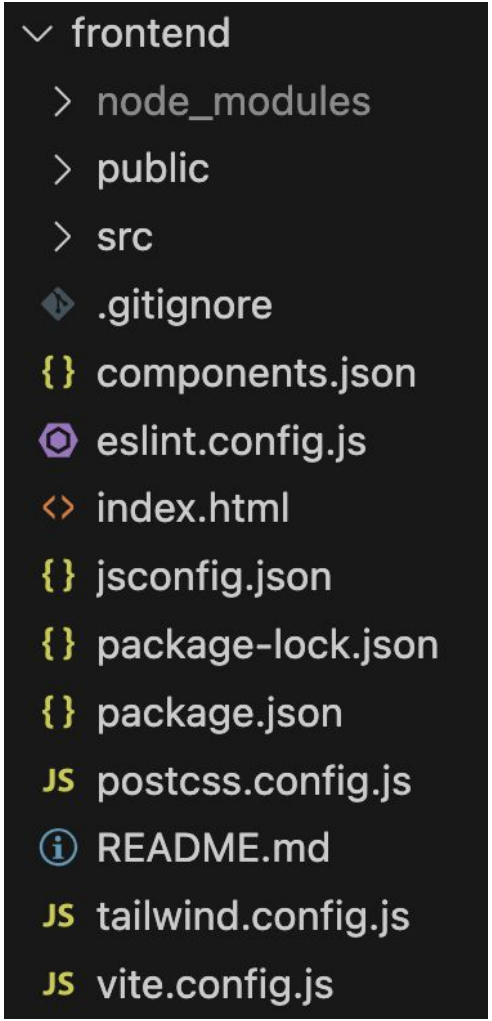
Project Structure



```
∨ JOBSCOUT
    > backend
    > frontend
```



```
∨ backend
  > node_modules
  ∨ src
    > config
    > controllers
    > middleware
    > models
    > routes
    > scrapers
    .env
    JS scraperManager.js
  .env
  .gitignore
  JS app.js
  {} package-lock.json
  {} package.json
  JS server.js
```



```
∨ frontend
  > node_modules
  > public
  > src
  .gitignore
  {} components.json
  eslint.config.js
  <> index.html
  {} jsconfig.json
  {} package-lock.json
  {} package.json
  JS postcss.config.js
  README.md
  JS tailwind.config.js
  JS vite.config.js
```

Backend

1. Data Scraping
   - Scrapes job postings from 6 different websites.

2. Data Processing
   - Cleans and organizes the scraped data.

3. Database Storage
   - Stores job listings in the database with fields like profession and location etc..

4. API Creation (MVC Model)
   - Sets up API endpoints for querying job listings based on parameters like profession and location (e.g., "Software Engineer" in "Helsinki").

Frontend

1. User Interface
   - Users interact via a clean, responsive UI with a search bar for profession and location.

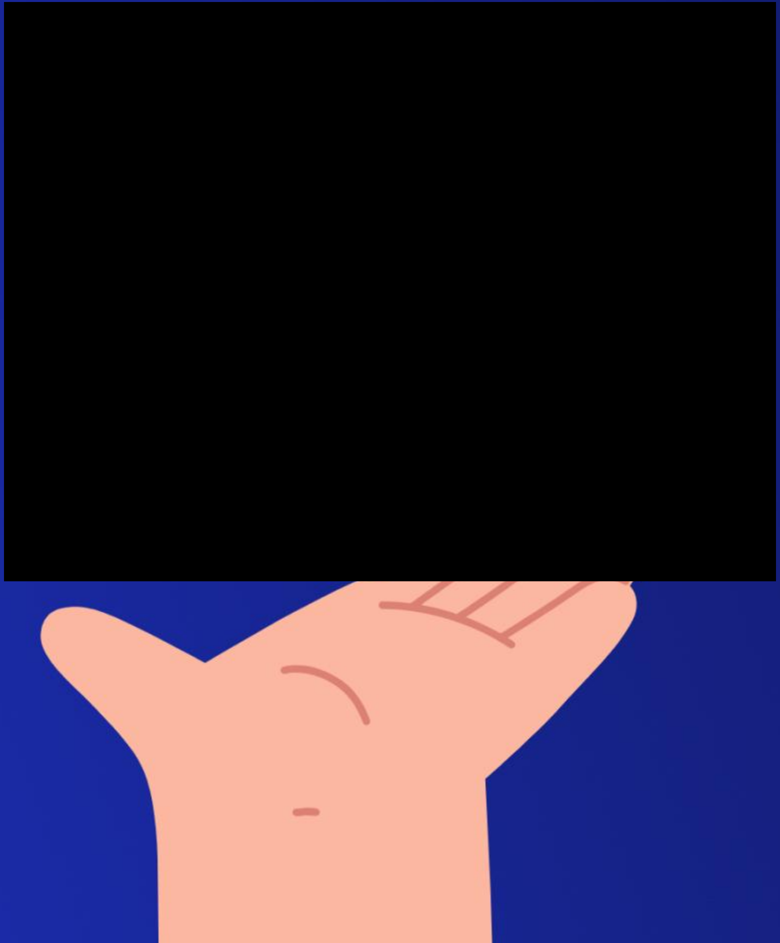
2. Login/Signup Modal
   - Users can sign-up and log in for personalized features.

3. Search Functionality
   - Users search for jobs based on profession and location.

4. Job Listings Display
   - Job listings (currently mock data) are shown based on the search.

5. Save Favorites
   - Users can save favorite job listings for future reference.

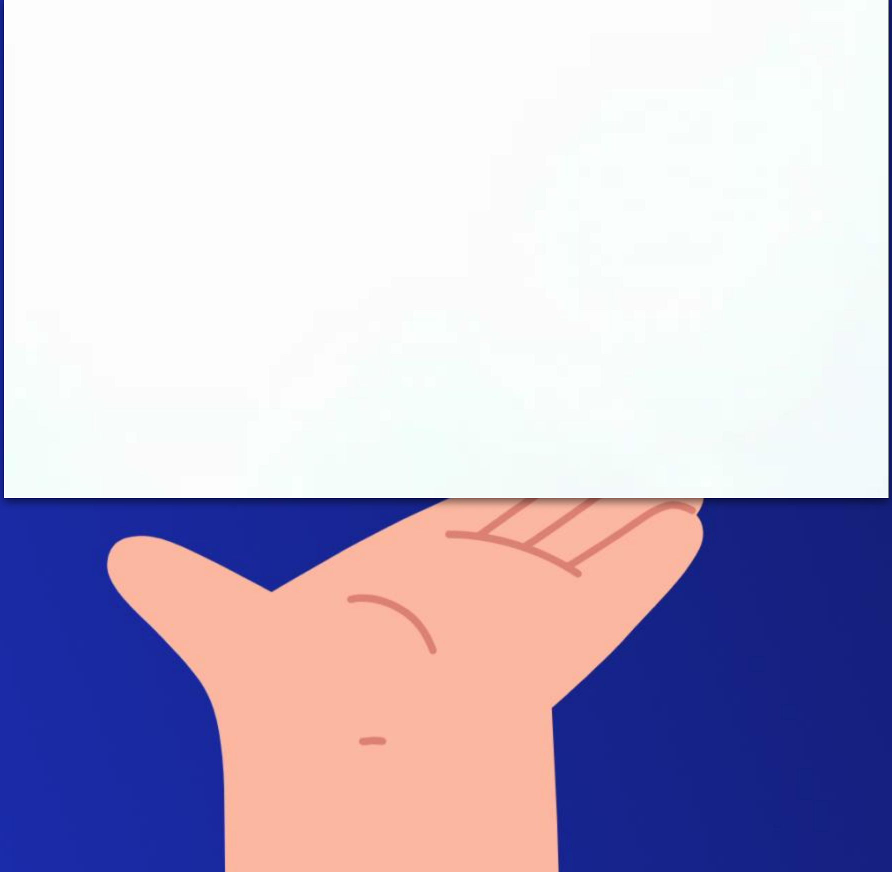
6. Current Status
   - The frontend uses mock data until backend integration is completed in Sprint 3.



Demo Of Our Backend

Scrapes and Saves to MongoDB Atlas

Frontend Demo



Demo Of Our Frontend

Sprint Ceremony

Daily Scrum

Ivan	Dev, Designer
Pavel	Product Owner
Viet	Dev
Saed	Sprint Master

We communicated in breakout rooms to ensure everyone was aligned. We also used Discord as communication channel.

Sprint Review

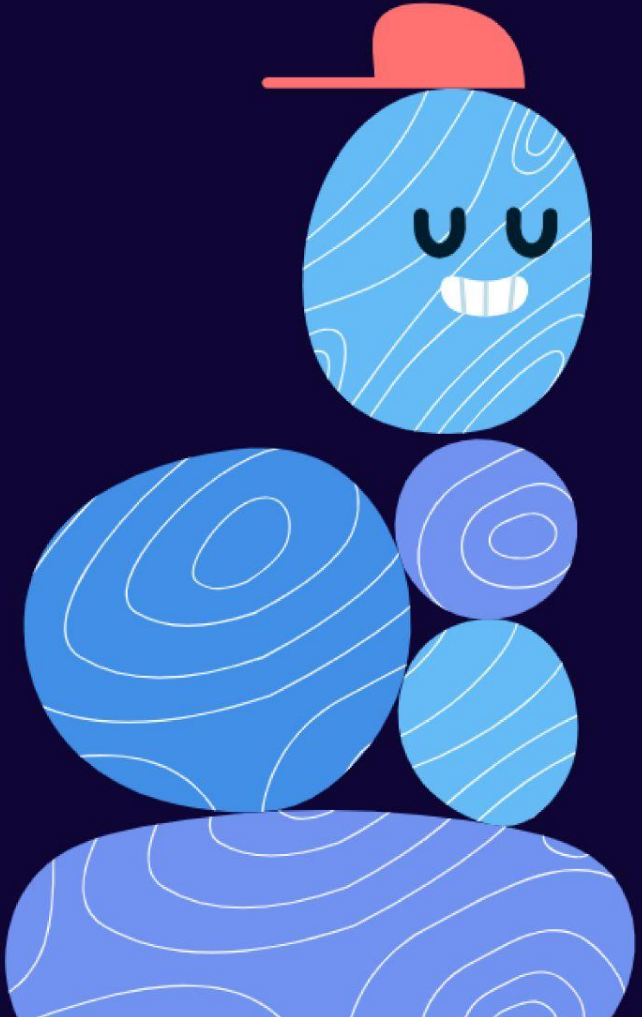
Sprint 2 was a success. The backend effectively scrapes and stores job data, while the frontend allows users to search and save job listings. Both demos were completed, and we’re on track for full integration in Sprint 3.

Sprint Retrospective

Successes:
Backend and frontend deliverables were completed.

Improvements Needed:
Accelerate backend-frontend integration and refine the existing functionalities.

Next Steps:
Complete integration in Sprint 3 and enhance user experience.



Questions?