Hunted

Access

- https://huntd.tech
- Android app
- iOS app

Project description:

Huntd is a comprehensive job search and application tracking app designed to assist job seekers in their quest for employment. With its user-friendly interface and powerful features, this app is essential for individuals actively seeking job opportunities.
Job seekers of all experience levels and industries can benefit from Huntd. Whether you're a recent graduate exploring entry-level positions or a seasoned professional looking for new career opportunities, this app provides the tools and resources necessary to streamline your job search process.
Huntd is particularly valuable for users who want to stay organized and efficient throughout their job search journey. With features like personalized job recommendations, application tracking, and a resume builder, this app ensures that users never miss an opportunity and can effortlessly keep track of their progress.

Out-of-scope features:

- Admin functionality;
- Question/feedback form.

Modules

1. Main page

For companies

The main page landing. Contains info for recruiters about our product. This page should motivate recruiters to go to the list of candidates and sign up.
The page should contain a comparison with other resources, quotes from the CEO, and logos of partners.

For engineers

Default main page for all users. Should contain a form for registration by email and social networks (Google, LinkedIn, GitHub), and links to Jobs and Web3 companies.
Authorized users should have a banner with the ad of the mobile app instead of a sign-up form.
At the very bottom of the page, there should be at least 10 feedbacks from real users.

2. Candidates list

The list of candidates should look like a list of cards with info about a candidate.

Filters

Filters could be seen by any user, but only authorized users should be able to use them (unauthorized should be able to sign in/sign up by hovering over the filters section).
Available filters should be:

- Role;
- Technologies;
- Desired base salary (annual/monthly);
- English level;
- Candidate's location (Timezone/ Country and City).

Candidates and Candidate profile

Each candidate card should contain a short description of the candidate's skills and achievements. Click on the “Show experience” should open all candidate’s experiences.
The recruiter should have the ability to start the chat with a candidate, and the candidate should be warned that he has to create a recruiter’s profile before chatting with candidates.

After clicking on the candidate’s card, the candidate’s profile should be opened in a new tab.
The candidate’s profile should contain all information about the candidate, but contacts with the candidate should be hidden before the candidate opens it (available only after the chat with a candidate).
Candidate’s experience should be sorted from the oldest to the newest positions.

3. Sign Up

After entering the email and the password, there should be two options for how to continue registrations: as a recruiter or as a candidate.
Available APIs for registration: Google, LinkedIn, GitHub.

As a recruiter

During the registration process, the recruiter should be able to fill his position, the company, and his contact information.

One of the main stages should be filling in the information about the required candidates:

- the role of the candidate;
- required technologies;
- salary (annual/monthly);
- years of experience;
- English level;
- Timezone or country/city of the candidate.

At the end of the registration process, the recruiter should be redirected to the list of candidates.

As a candidate

Stages of registration:

1) Role: info about desired position, tech skills (5-15);
2) Expectations: info about the experience, expected salary (annual or monthly, dollar or euro), English level, and location;
3) Experience: detailed information about experience, should be possible to export from LinkedIn;
4) Bio: info about the achievements of the candidate and expectations from work;
5) Contact Information: avatar, full name, CV to upload, and links to social networks.

After successful registration, users should wait for profile activation by Admins during the next 24-48 hours.

4. Sign In

The Sign In page should have the “Sign up” link which redirects to the “Sign up” page, and the “Forgot the password” link which redirects to the “Forgot password” page.

5. Chats

Recruiters should be able to initiate a chat with a candidate on the list of candidates and on the candidate profile page.
Before contacting the candidate in the chat, the recruiter shouldn’t be able to see the contacts of the candidate.
The candidate should be able to open their contacts or decline the recruiter’s proposition and not share contact details.
The recruiter should be able to send the offer to the candidate or mark the chat as rejected.
To group chats, candidates and recruiters should have the ability to Archive chats or mark them as Favourite.

6. Profile

Users should be able to have candidate and recruiter accounts simultaneously.
After the registration, the user should be able to create a profile of the recruiter or the candidate.
In the profile section, the user should be able to:

- Edit his profile;
- Switch between recruiter/user profiles;
- Connect social networks (LinkedIn, GitHub, Google);
- Change password.

Users should be able to activate or deactivate their candidate profile in the profile settings.
Admins should be able to edit user profiles at any time.

7. Footer

The footer should contain the following:

- “TOP 100 WEB3 COMPANIES” with a preview of the top 5 companies;
- 3 columns of vacancies for web3 developers;
- links to social networks (LinkedIn, Twitter, Telegram, Signal, Instagram, Facebook);
- links to documents, pricing, FAQ, and About us.

8. Web3 companies and Jobs

This page should contain a table with 100 web3 companies split by 10 like:

- 10/100;
- N/100;
- 100/100.

Each company should have a clickable logo and name. Clicking on it should open a page in a new tab with a list of vacancies for this company.
Authorized users should be able to post a new job manually or using import from ATS.
Applying with 1 click should be available for authorized users.

To filter vacancies users should have filters with skills.

Unauthorized users should only be able to observe some vacancies and after clicking on [View more] should be suggested to sign in.

At the very bottom of the page users should have the ability to subscribe to vacancies using the form with the fields:

- Desired roles;
- Experience;
- Email.

Mobile app

The mobile app is an MVP to make communication easier and, you know, more mobile.
The app contains only some features of the web app e.g. some chat features, IAM, and profile settings.
Test only implemented features and do not create a feature/improvement requests for the mobile app as they will be rejected.

Tasks

for Flex students

1. **Test plan**. Prepare a joint test plan for the web application (desktop and mobile web browser) and the mobile application (Android or iOS).
2. **Decomposition** (for browser and mobile applications) to cover test cases.

   For the web application and the mobile application, you need to create two separate projects.

3. **Integrate** Jira + TestRail:
   - Create TestRail and add part-time mentors;
   - Create a new **project** in Jira.
4. Test design:
   - permission testing table;
   - RTM (example).
5. Prioritize your tasks.
6. Transfer documentation:
   - create tickets in Jira - one story per feature. Each story must have the feature requirements described in this document;
   - test the features: read the features description, and test the application against the requirements (but **do not limit it to the existing requirements**, because there are always implicit requirements).
7. Web features testing: cover all features with test cases in TestRail, test according to test cases, and report found bugs.
8. Mobile features testing: cover all **implemented** on the mobile app features with test cases in TestRail (do not cover with test cases and do not create Jira tickets for features, which were not developed in the app). Test according to test cases, and report found bugs. You should test either the Android or iOS app.
9. Write a test report at the end of practice. In the Test Summary spreadsheet, provide statistics on the execution of test cases and found bugs.

Tasks

for Full-Time students

1. **Test plan**. Prepare a joint test plan for the web application (desktop and mobile web browser) and the mobile application (Android or iOS).
2. **Decomposition** (for browser and mobile applications) to cover test cases.

   For the web application and the mobile application, you need to create two separate projects.

3. **Integrate** Jira + TestRail (scrum master):
   - Create TestRail and add all participants;
   - Create a new **project** in your team's Jira.
4. Test design:
   - permission testing table;
   - RTM (example).
5. Prioritize and distribute tasks together with the team.
6. Transfer documentation:
   - create tickets in Jira - one story per feature. Each story must have the feature requirements described in this document;
   - test the features: read the features description, and test the application against the requirements (but **do not limit it to the existing requirements**, because there are always implicit requirements).
7. Web features testing: cover all features with test cases in TestRail, test according to test cases, and report found bugs.
8. Mobile features testing: cover all implemented on the mobile app features with test cases in TestRail, test according to test cases, and report found bugs. You should test either Android or iOS apps (assuming there are people on the team who can do it). Emulators are allowed.
9. Write a test report at the end of practice. In the Test Summary spreadsheet, provide statistics on the execution of test cases and found bugs.

Explanation

for Full-Time & Flex students

1. Perform test design with the following points:
    - **Permission testing table**: user logged out/recruiter/candidate.
    - **RTM**: you can start creating a matrix in the first days, but fill it in after writing the test cases.
2. Create separate stories for the tasks you are working on (test plan, RTM, etc.)
3. Before moving tasks from the backlog to the sprint, you should add a description to each task and identify the person responsible for that task.
4. Your Jira scrum board can have multiple statuses:
    - **To Do**: a task is created, contains a functional description, is a responsible person, and is ready for processing.
    - **In Progress**: you work on the task and describe functionality with test cases.
    - **On Review**: a task is described by test cases and needs a review before starting testing. If it is a task for, for example, creating a test plan — after the review, it is transferred to Done.
    - **Testing**: testing the task by test cases and reporting bugs.
    - **Done**: task was tested (all test cases passed, bugs reported).

Bugs found during testing should be included in the product's Jira backlog and linked to the corresponding story. Describe bugs using all applicable fields.

5. Create separate projects in TestRail and RTM for web and mobile test cases.

Do not forget to create and conduct a test run:

- One for each feature;
- Smoke for the entire site (no need to run);
- One test run for all test cases on the web (no need to run);
- One test run for all test cases on mobile.

Test runs with all test cases for web and mobile should be attached to the report at the end of the practice. The practice report should contain links to your Jira, TestRail, RTM, Test Plan, and statistics with completed test runs. After writing all test cases, do not forget to **export**.

6. **How to submit:** each team member must have a Story with their name (e.g. [Hunted testing]: Jon Snow), which they send to the review on the platform. The Story should have:

- test report;
- listed tasks on which they worked;
- linked stories they tested;
- linked found bugs.

Plan

for **full-time** students only

Day 1: Familiarization with the project:

- Familiarize yourself with the functionality of the project;
- Set up the necessary testing tools and access to Jira/TestRail;
- Distribute tasks among team members;
- Create stories in Jira with tasks for each team member;
- Start creating the necessary test documentation;
- Set up calls for 4 days.

Day 2: Writing and reviewing test cases and test documentation:

- Describe functionality in your area of responsibility with test cases;
- Review the test cases of colleagues;
- Monitor the status of writing test documentation (test plan, RTM).

Day 3: Conducting testing and reporting bugs.

Day 4: Completion of test activities, writing a test report.

Calls

for **full-time** students only

- **10:00** Daily call: status update, analysis of tasks for the day.
- **14:00** Re-sync with task update and discussion of blockers. Don't spend more than 30 minutes on the call, discuss only actual tasks and problems.
- **17:00** Final sync. Discuss what you managed to do today and what needs to be done tomorrow.