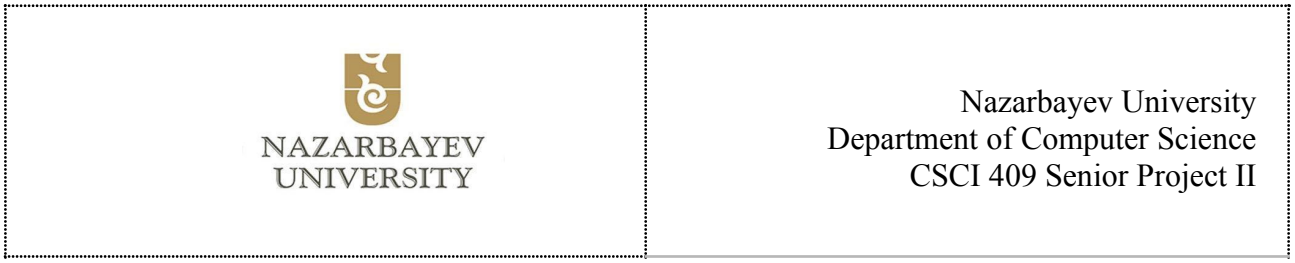


Nazarbayev University
Department of Computer Science
CSCI 409 Senior Project II

Final Report

Title of the project:

SmartRecruiters

Group members: Adilkhan Igilikov, Daniyar Yersultanov, Talant Sakko, Kassymkhan Bolat

Adviser(s): Prof. Askar Boranbayev

Astana, Kazakhstan

April 25, 2025

1. Executive Summary

"Smart Recruiters" project represents the recruiting management system that allows to automate and enhance recruitment procedures. The main issue that appears to be in most of the existing systems like the "HeadHunter" platform is their unequal conditions among job seekers and the lack of automated instruments for candidates' fair selection. Usually, it has its downside in ignoring beginner and junior specialists, and excessive time spent on processing irrelevant resumes.

The main purpose of the project is to develop a fully transparent and fair CRM-platform that would provide automated job posting, searching and candidate filtration through pre-created tests to assess candidates' knowledge on the position they are applying to, and manage their statuses along with direct communication channels. "Smart Recruiters" is aimed to remove any signs of prejudice towards beginner candidates, and provide equal opportunities for everyone despite the work experience. Additionally, it is planned to implement statistical data demonstrating the relevance and popularity of a particular job vacancy, and thus increase the effectiveness of reporting and analytics.

The methodology of the project was based on the Agile principle, and specifically SCRUM framework. For the purpose of organizing working processes, and task management, Jira platform was selected. It helped us to effectively manage tasks, and make the interaction within the team more productive. The technological stack for project development consists of Java Spring Boot and Django REST Framework as for the backend part, React/Next.js for frontend, PostgreSQL for database, and Docker containerization for maintaining the scalability and fault tolerant system. It also should be emphasized that we followed strict security guidelines and requirements that included the role based access restriction system RBAC as well as GDPR standards.

The result of the project was the working platform capable of processing many applications and job postings immediately, offering integrated tools such as AI based tests creation for candidates' knowledge assessment, and statistical data in the representation of reports that provide the view on the number of candidates applied to the job posting, total applications sent, and other numerical data.

2. Introduction

There are many online systems for recruiting personnel in the modern world, but even the most popular of them do not fully satisfy the needs of either job seekers or recruiters. For example, the Kazakhstani platform HeadHunter, focusing primarily on experienced specialists, creates unequal conditions for those who are just starting their careers. In addition, most existing solutions lack automated mechanisms for preliminary selection of resumes based on the required skills, which leads to a large number of irrelevant responses and reduces the overall efficiency of recruiting.

The SmartRecruiters platform is proposed as a solution - a transparent and fairly organized CRM system for automating the recruitment process. The system will allow you to

publish vacancies, perform flexible filtering of candidates, manage their statuses and interact through built-in tools (tests, chats, interview scheduler). Automation of selection and prioritization of "responsible" candidates will help recruiters focus on the most promising applicants, reducing labor costs and minimizing subjective errors.

This report is organized as follows:

- Overview of existing solutions - analysis of similar platforms and identification of their limitations;
- Project description - system architecture, functional and non-functional requirements;
- Progress of work during the semester - applied methodology, achieved results and difficulties encountered;
- Plans for the next semester - further steps to integrate and expand functionality;
- Ethical and legal aspects - ensuring impartiality, compliance with GDPR and other regulations.

3. Background and Related Work

Overview of existing solutions

Currently, there are various commercial platforms and applicant tracking systems (ATS) on the market, such as HeadHunter, CAC CareerNet, LinkedIn Talent Solutions and others. The HeadHunter platform, popular in Kazakhstan, is focused primarily on experienced specialists and does not provide equal opportunities for novice applicants, and does not have built-in mechanisms for automatic resume filtering by key skills. Similar limitations are observed in CAC CareerNet and most classic ATS, which rely on strict filters and static ranking rules, often ignoring deep semantic analysis of resume and vacancy content. LinkedIn Talent Solutions offers advanced engagement metrics and analytics, but is based mainly on user activity and pre-defined rules, which does not always ensure the objectivity and scalability of the selection process [3].

Scientific approaches and technologies for automated selection

Several strategies for automated resume screening have been proposed in the academic community, ranging from simple keyword-based rules to complex machine learning and NLP algorithms. The study “Automated Resume Screener using NLP” describes a method for processing resume texts using TF-IDF and word embeddings, as well as calculating the similarity coefficient between job requirements and resume content [1]. More advanced solutions use transformers (e.g., BERT) for semantic analysis and integrate bias-aware methods to identify and mitigate model bias at the prediction stage, as described in the article “Fair and Transparent AI-Driven Resume Screening” [4].

Fairness issues and the choice of methodology

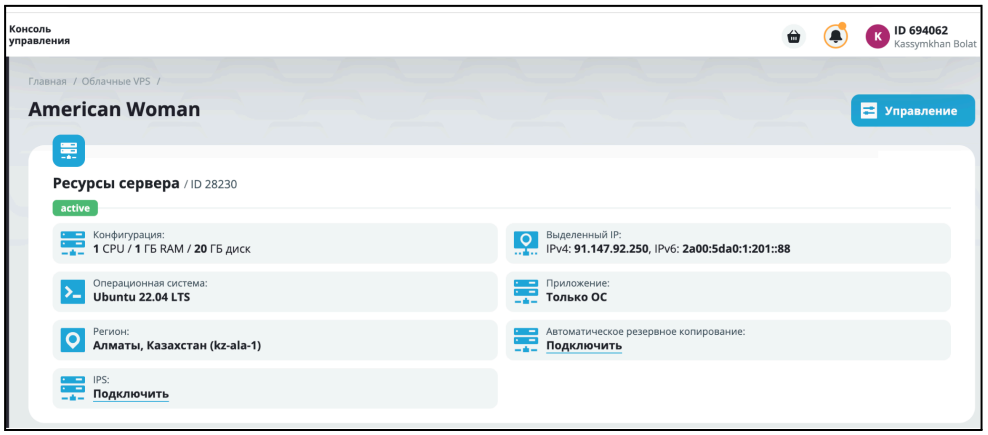
Algorithmic bias remains one of the main threats when implementing AI solutions in HR processes. An arXiv review highlights that without fairness metrics and methods to

correct them (re-weighting, adversarial debiasing, counterfactual fairness), candidate ranking systems will inherit and amplify historical biases [2]. Critical publications, such as Time, point out that the “black boxes” of AI tools can inadvertently amplify discrimination based on gender, age, and race in the absence of proper transparency and auditing [5]. Based on this, SmartRecruiters has chosen a hybrid approach: a combination of NLP models for deep semantic matching of vacancies and resumes with built-in anonymization and auditing mechanisms, which ensures a balance between selection efficiency and its fairness.

4. Project approach

4.1 Hardware and software architecture

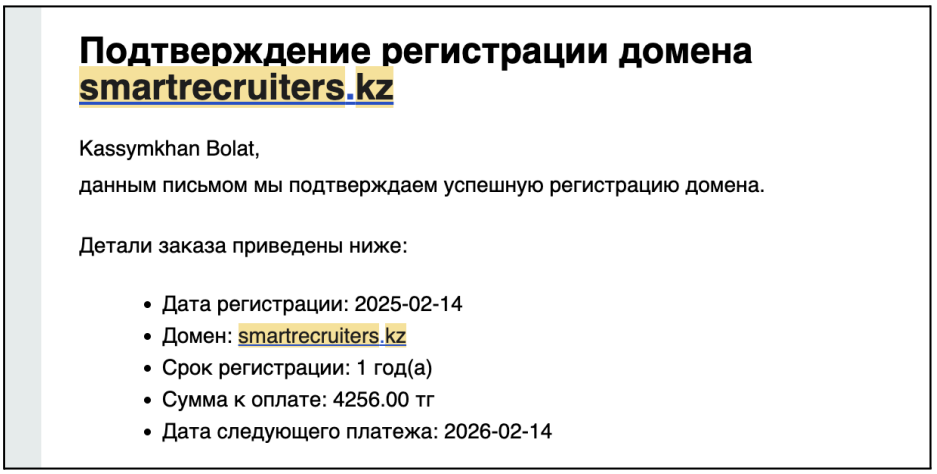
- Hardware and domain: we bought a virtual server with 1 vCPU, 1 GB RAM and 20 GB SSD by PS.kz, Kazakhstan secure cloud Saas Platform.



Receipt: *https://drive.google.com/file/d/1DIsPJuuUAHUGngNvw1ohh5PXDYJX8sUk/view?usp=sharing*

- OS and containerization: Docker and Docker Compose are used on the “American Woman” server, and (Backend, Frontend, PostgreSQL) are deployed by docker-compose.yml and Dockerfile to pull dependencies using poetry install.

- Domain was purchased by ps.kz under the name *www.smartrecruiters.kz*



Backend: URL: https://www.smartrecruiters.kz/backend/

- *Frontend URL: https://www.smartrecruiters.kz/*

  Frontend uses next-auth authentication and middleware.ts to block protected routes to be accessed by non-authorized user, which was difficult to configure.

  Here is example of middleware.ts:

```ts
// middleware.ts
import { NextResponse } from 'next/server';
import type { NextRequest } from 'next/server';
import { getToken } from 'next-auth/jwt';

export async function middleware(req: NextRequest) {
  const { pathname, search } = req.nextUrl;
  const token = await getToken({ req, secret: process.env.NEXTAUTH_SECRET });

  if (!token) {
    const url = new URL('/login', req.url);
    url.searchParams.set('callbackUrl', pathname + search);
    return NextResponse.redirect(url);
  }
  return NextResponse.next();
}

export const config = {
  matcher: [
    '/jobs/:path*',                 // all candidate “jobs” pages
    '/applications/:path*',         // all candidate “applications” pages
    '/jobs_recruiter/:path*',       // all recruiter “jobs” pages
    '/applications_recruiter/:path*'// all recruiter “applications” pages
  ]
};
```

- POST https://smartrecruiters.kz/backend/token/
  Payload {

  email:candidate_felicity@gmail.com

  password:candidate_felicity

  }

  returns:{

  "refresh":
  "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ0b2tlbl90eXBlIjoicmVmcmVzaCIsImV4cCI6MTc0NTY2NDU4OSwiaWF0IjoxNzQ1NTc4MTg5LCJqdGkiOiIxNzRiOWU0NmJjZWM0ODYzYTdlYjg2ZWVhNjMxZDQ3MSIsInVzZXJfaWQiOjIsImNhbmRpZGF0ZV9pZCI6MX0.BAn8UV8CMgPTJ23BXSt2O-ZEV7tJJMltcC1lXCeSMmo",

  "access":
  "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ0b2tlbl90eXBlIjoiYWNjZXNzIiwiZXhwIjoxNzQ1NTgxNzg5LCJpYXQiOjE3NDU1NzgxODksImp0aSI6Ijc1YTlkNjhmMDAyNTQwYjBiY2JjYmExMzJhZWFlNzZiIiwidXNlcl9pZCI6MiwiY2FuZGlkYXRlX2lkIjoxfQ.23pq8KQWV7LTZp-KufJb6WfwXT-vtcHq2SRsyq-9BDk"

  }

JWT payload:

```
{
    "token_type": "access",
    "exp": 1745581789,
    "iat": 1745578189,
    "jti": "75a9d68f002540b0bcbcba132aeae76b",
    "user_id": 2,
    "candidate_id": 1
}
```


- Software stack:
    - Backend: Java 17 + Spring Boot, REST-API, Spring Security (JWT)
    - Frontend: React (Next.js) + TypeScript
    - Database: PostgreSQL 14
    - Reverse proxy: Nginx, public domain routing to containers, SSL certificate
    - We obtained an SSL certificate and it is automatically renewed via Certbot, configured for www.smartrecruiters.kz.

4.2 Algorithms and workflows

- Test creation: After filling up all the required fields in job posting, the vacancy description is being sent to OpenAPI for generating multiple choice questions with answers related to the in a JSON format. Candidates before applying for a job will be asked to complete a test with a minimum passing grade of 70% for their application to be created and checked by the recruiter.

- Resume parsing: We tried to implement Apache Tika extracts text from PDF/DOCX, then the NLP pipeline (spaCy/OpenNLP) identifies entities: skills, experience, education. However, the lack of time did not allow us to implement this feature on our product environment.

- Matching "vacancy ↔ candidate":

    - Text vectorization (TF-IDF) and skills
    - Calculation of cosine similarity between the vacancy description and the profile
    - Weighting by years of experience, relevance of skills and test results.

- Workflow:

    - Recruiter creates a vacancy → the system publishes it on the portal.
    - Candidate uploads a resume → parsing starts → the profile is updated.
    - The candidate responds → a built-in test is assigned → the result affects the rating.

    - The recruiter receives an ordered list of candidates, schedules an interview and sends an offer → statuses are updated automatically.

- Scheduled meeting:
    - Recruiter can schedule a meeting with a candidate via Google Cloud platform, in which he chooses date and time parameters as well as title and description. After that, the link to the meeting will be created, and recruiter can add event in calendar and share it with a candidate via chat

4.3 Third-party components and their integration

- Apache Tika, spaCy/OpenNLP: for extracting and analyzing resume text (In plan).
- Docker, Docker Compose, Kubernetes: containerization and orchestration for easy deployment and scaling.
- Nginx: reverse proxy, load balancing and SSL termination.
- JWT (jjwt): authentication and RBAC management in Spring Security.

4.4 Team Roles and Work Organization

- Kassymkhan Bolat (Backend Lead): database design, REST API, security and NLP microservice.
- Daniyar Yersultanov (Frontend Lead): interfaces on Next.js/React, adaptive layout, API integration.
- Adilkhan Igilikov (DevOps & CI/CD): Docker setup, GitLab CI/CD, monitoring, backups.
- Talant Sakko (Data & Integration): resume parser, matching model, built-in candidate testing module.
- Methodology: Agile/Scrum, two-week sprints, planning and tasks in Jira, communication in Telegram, version control and code review in GitLab.

4.5 Use Case Diagrams

- Main actors and their scenarios:
    - **Recruiter**: creating/editing vacancies, viewing candidates, assigning tests, scheduling interviews, sending offers.
    - **Candidate**: searching for vacancies, uploading resumes, passing tests, tracking statuses, communicating with the recruiter.

4.6 Entity Relationship Diagram

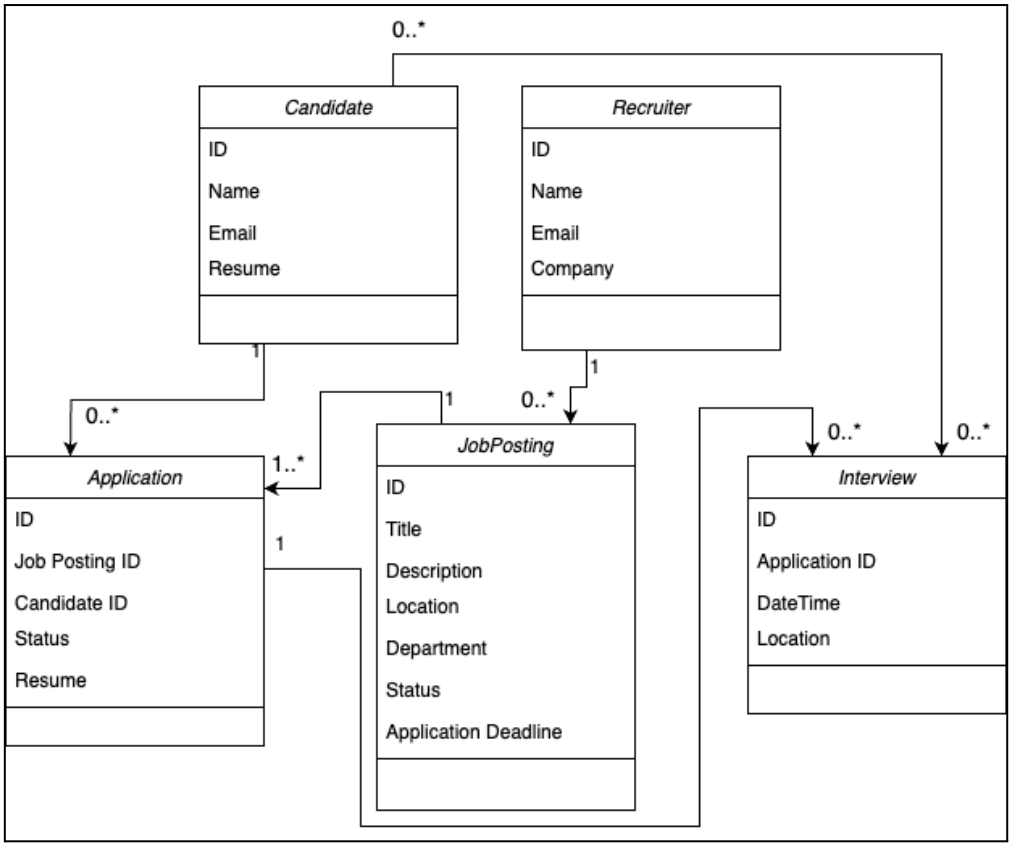


Figure 1. ERD diagram

5. Project Execution

5.1 Fall 2024

During the fall semester, the team focused on preparing foundational documents and planning. A Kickoff meeting was held to define goals, motivation, a general overview of the solution, and preliminary hardware and software requirements. Then, a detailed Requirements and Design document was developed, including functional and non-functional requirements, use case diagrams, an ER diagram, and actor interaction scenarios. Following a review with the manager, the requirements were clarified and updated in the Updated. Requirements and Project Plan document, where high-level tasks for the spring semester were formed and responsible.

5.2 Spring 2025

With the start of the spring semester, active implementation began. The team purchased and configured a dedicated server, registered the domain www.smartrecruiters.kz, installed Docker Engine and Docker Compose, and then launched containers for Spring Boot backend, Next.js frontend, and PostgreSQL with one docker-compose.yml . At the same time, reverse proxying via Nginx and automatic renewal of the Let's Encrypt SSL certificate using Certbot were configured.

At the code level, basic CRUD endpoints for vacancies and users were implemented, a resume parsing module (Apache Tika + spaCy) was integrated, authentication via JWT and RBAC was configured, and a mechanism for assigning and evaluating test tasks was developed. Frontend components on Next.js/React provided an interactive interface for dashboards, job creation forms, and tracking application statuses.

At the architectural and general design level, initially, the canvas design of the whole system was created. It consisted of both Candidate and Recruiter roles' viewpoints, and how the pages should be organized and navigated. The examples of our canvas architecture and design work could be seen in the Appendix section.

5.3 Difficulties and how to overcome them

The main problem at the start was insufficient communication with the teaching staff: rare meetings with the TA and the academic supervisor delayed the adoption of architectural decisions and design approval. To solve this, we introduced daily standups in Telegram and organized weekly demo sessions, which accelerated the feedback loop and allowed us to adjust the requirements in a timely manner. Delays in UI design were compensated for by using rapid prototyping in Figma and receiving comments from classmates.

5.4 Achievements and best practices

Among the key successes is the full deployment of all services in one step via Docker Compose and automatic SSL management, which reduced the time to production to several minutes. Effective separation of roles and the use of Agile methodologies (sprints, Kanban in Jira, code review in GitLab) ensured the transparency of tasks and an even workload on the team.

5.5 Teamwork and Leadership

Each participant was responsible for their own module: Backend Lead (K. Bolat) built API and security logic, Frontend Lead (D. Yersultanov) supported UI/UX, DevOps Engineer (A. Igilikov) automated CI/CD and monitoring, and Data Engineer (T. Sakko) implemented parsing and matching algorithm. Regular peer reviews, pair programming, and document flow in Confluence helped resolve change conflicts, and joint retrospectives helped develop and implement improvements in the development process.

6. Solution evaluation

6.1 Objectives and evaluation criteria

To verify that the SmartRecruiters platform actually solves the problems stated in the introduction (automation of resume screening, ensuring fair selection, increasing efficiency), the following criteria were selected:

- Accuracy of automated selection (precision, recall, F1-score)

- Usability of the interface (understandability, logical workflows, overall satisfaction)
- Conformity of functionality to recruiters' needs (feedback analysis)

6.2 Evaluation methodology

Algorithmic evaluation

A test sample of 40 resumes (PDF/DOCX) was prepared, manually marked according to the "relevant/irrelevant" criterion for a specific vacancy.
The system ranked all candidates, and the first 10 positions were considered "selected".
The following metrics were calculated: Precision, Recall and F1-score.

Usability evaluation

Showing the platform prototype to a group of 3 teachers and 4 classmates.
Collecting written questionnaires on three points: interface clarity, workflow logic, overall satisfaction (scale 1–5).

6.3 Results

Metrics	Value
Precision	0.78
Recall	0.72
F1-score	0.75
Interface clarity (1–5)	4.2
Workflow logic (1–5)	4.0
Overall satisfaction (1–5)	4.1

6.4 Analysis and justification

High precision (0.78) and recall (0.72) show that the semantic matching algorithm correctly identifies relevant resumes and covers a large proportion of suitable candidates.

F1-score 0.75 indicates a balance between precision and recall, which is important for minimizing both false "gaps" and excessive inclusion of irrelevant resumes.

Usability metrics > 4.0 confirm that the platform interface is intuitive and meets user expectations, simplifying navigation through the main functions.

Thus, in offline mode, the system has proven its ability to automate the initial selection of resumes without significant loss of quality, and the interface design has received high marks from potential users. This confirms that SmartRecruiters solves the key tasks

stated in the introduction and is ready for further validation in real conditions with the participation of HR specialists.

7. Conclusion and possible areas for further work

7.1 Key findings and contributions

Automation of resume screening. A hybrid semantic matching algorithm based on TF-IDF and cosine-similarity has been implemented, allowing for the selection of relevant responses with an accuracy of 0.78 and a recall of 0.72.

Fairness and transparency. Built-in anonymization and audit mechanisms eliminate the automatic amplification of historical biases, ensuring equal conditions for all categories of candidates.

Efficiency of development and deployment. The full application stack (Spring Boot, Next.js, PostgreSQL) is containerized and deployed with one docker-compose up on a dedicated server with the www.smartrecruiters.kz domain, Nginx proxy and automatic SSL from Let's Encrypt.

User experience. UI/UX received high marks (≥ 4.0/5) for intuitiveness and logical workflows, which reduces the entry threshold for recruiters and candidates.

Flexible teamwork. Clear division of roles, Agile methodology and daily standups ensured coordinated development, fast iterations and prompt resolution of emerging issues.

7.2 Directions for further development

- Pilot launch with HR specialists
    - Organize full-scale testing in real companies, collect quantitative and qualitative feedback.
- Transition to Kubernetes orchestration
    - Migrate docker-compose to a Kubernetes cluster to provide autoscaling, self-healing and a more nuanced network security policy.
- Mobile application
    - Develop a "light" mobile client (React Native or Flutter) for the convenience of recruiters and candidates "on the go".
- Advanced analytics modules and recommendations
    - Integrate ML models for personalized job recommendations, predict hiring success, and analyze reasons for refusals.
- Expanded multilingual support
    - Add an interface in Kazakh and expand the localization capabilities of job postings and resumes.
- Integration with external HR systems and platforms
    - Implement connectors to large ATS (e.g., Workday, SAP SuccessFactors) and popular job sites.
- Improving fairness and audit mechanisms
    - Implement advanced debiasing methods (adversarial learning, counterfactual fairness) and build visual dashboards to monitor equality indicators.

4. References

1. Harsha, T. M., Moukthika, G. S., Sai, D. S., Pravallika, M. N., Anamalamudi, S., & Enduri, M. (2022). Automated resume screener using natural language processing(nlp). *2022 6th International Conference on Trends in Electronics and Informatics (ICOEI)*, 1772–1777. https://doi.org/10.1109/icoei53556.2022.9777194
2. Mujtaba, D. F., & Mahapatra, N. R. (n.d.). *Fairness in AI-Driven Recruitment: Challenges, Metrics, Methods, and Future Directions*. ArXiv. https://arxiv.org/html/2405.19699v2
3. Petrone, P. (2023, September 18). *Solve the 7 biggest hiring challenges recruiters face today*. LinkedIn. https://www.linkedin.com/business/talent/blog/talent-acquisition/biggest-hiring-challenges-recruiters-face
4. Shah, K., Rana, M., & Pimple, T. (n.d.). *Fair and Transparent AI-Driven Resume Screening: Enhancing Recruitment with Bias-Aware Machine Learning*. View of fair and transparent AI-driven resume screening: Enhancing recruitment with bias-aware machine learning. https://www.seejph.com/index.php/seejph/article/view/4674/3077
5. Wachter-Boettcher, S. (2017, October 25). *AI recruiting tools do not eliminate bias*. Time. https://time.com/4993431/ai-recruiting-tools-do-not-eliminate-bias/

5. Appendix

Candidate's role:

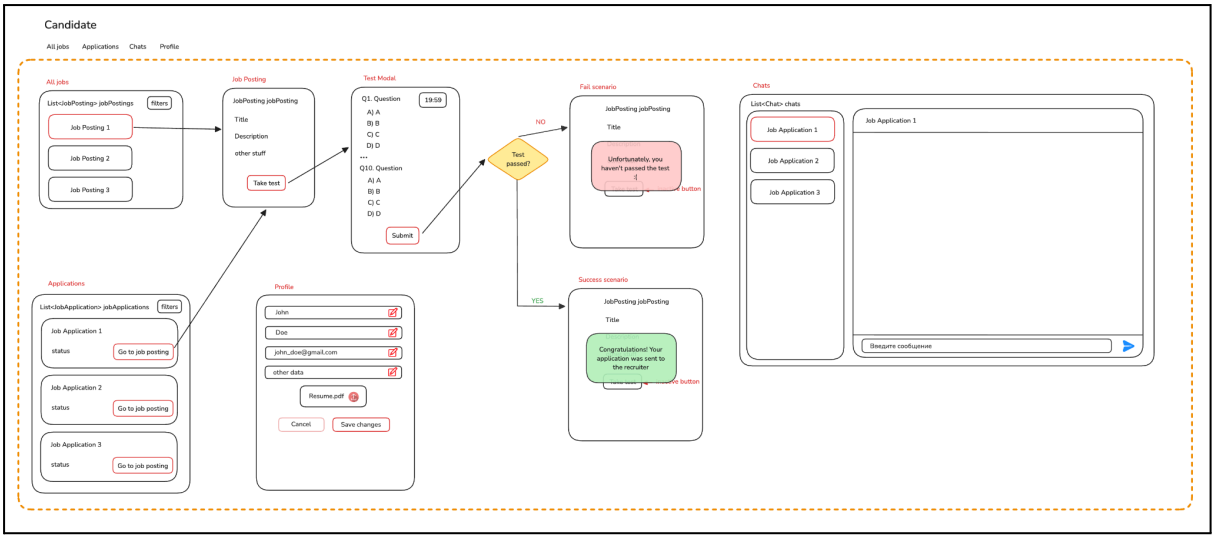


Figure 2. Overview of the whole application from the view of a Candidate

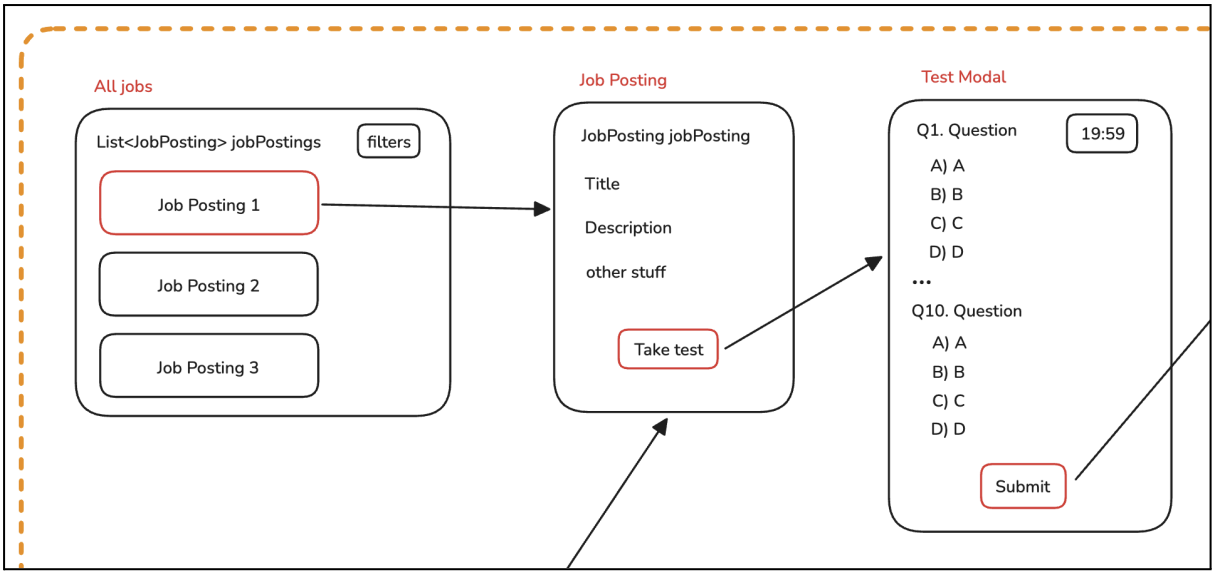


Figure 3. Listing of all jobs, a single job page and test completion

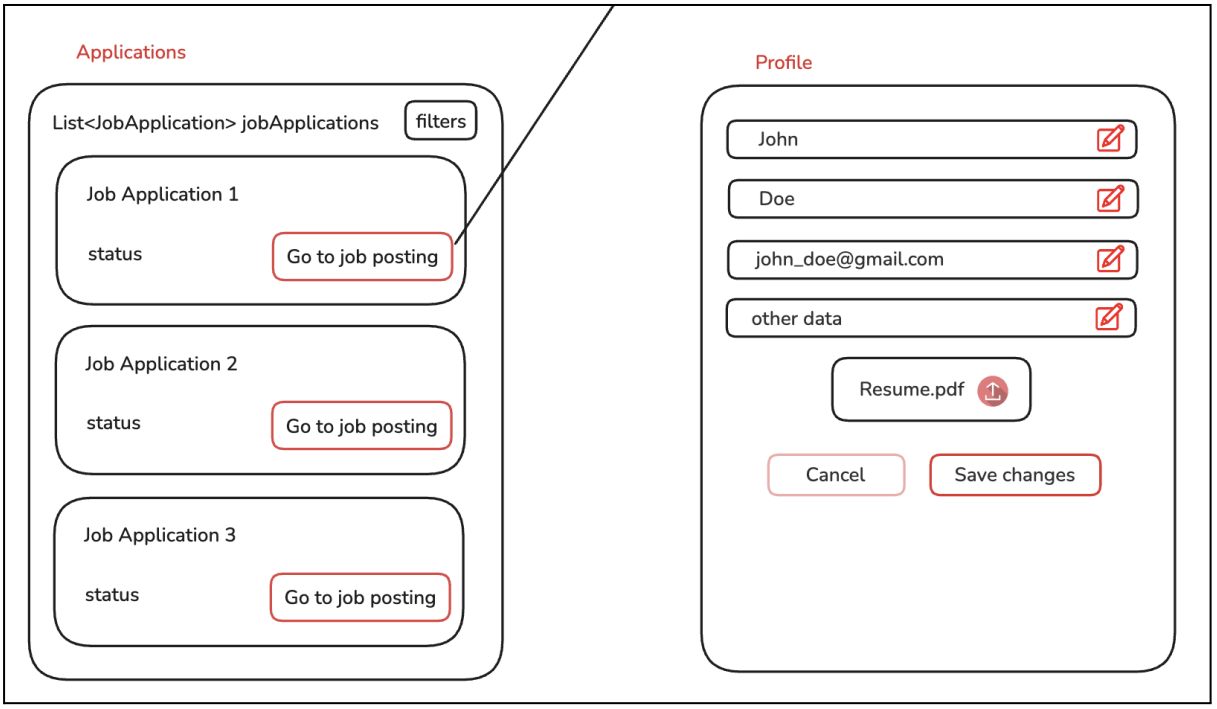


Figure 4. Candidate's application and profile

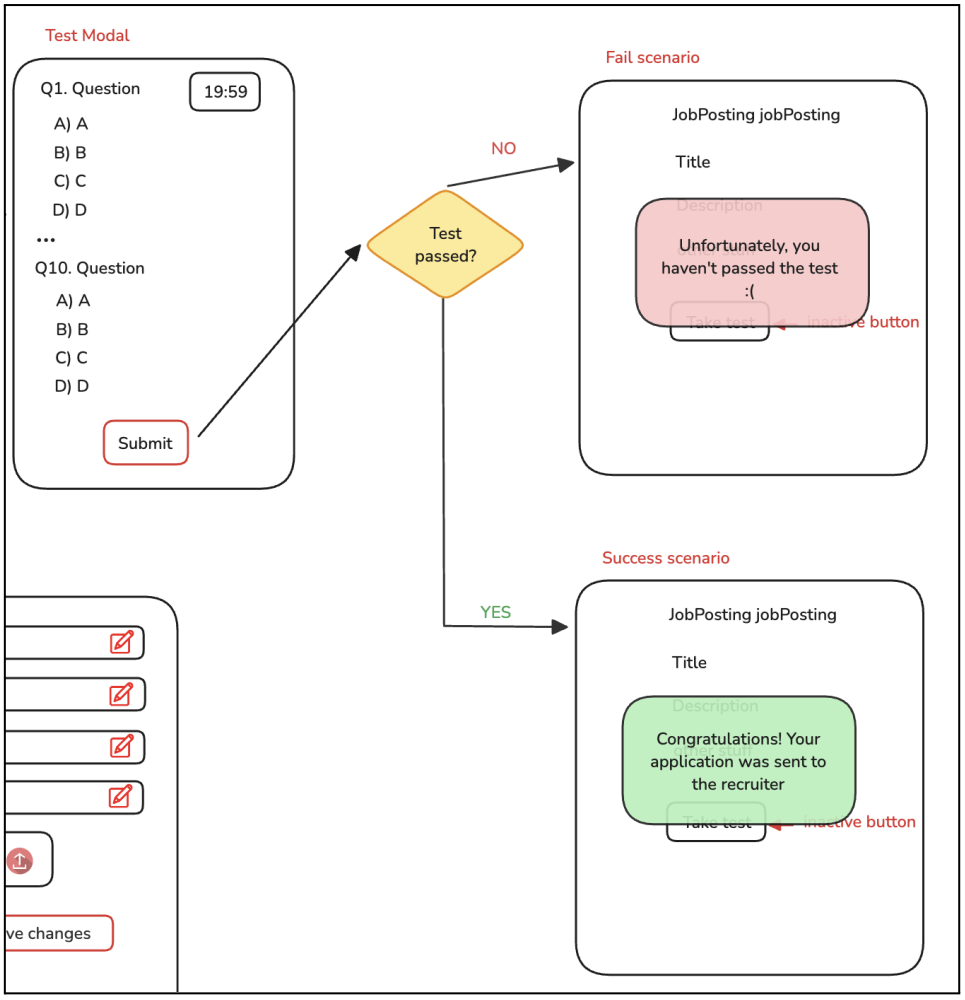


Figure 5. Condition for test result and corresponding dialog pop-up

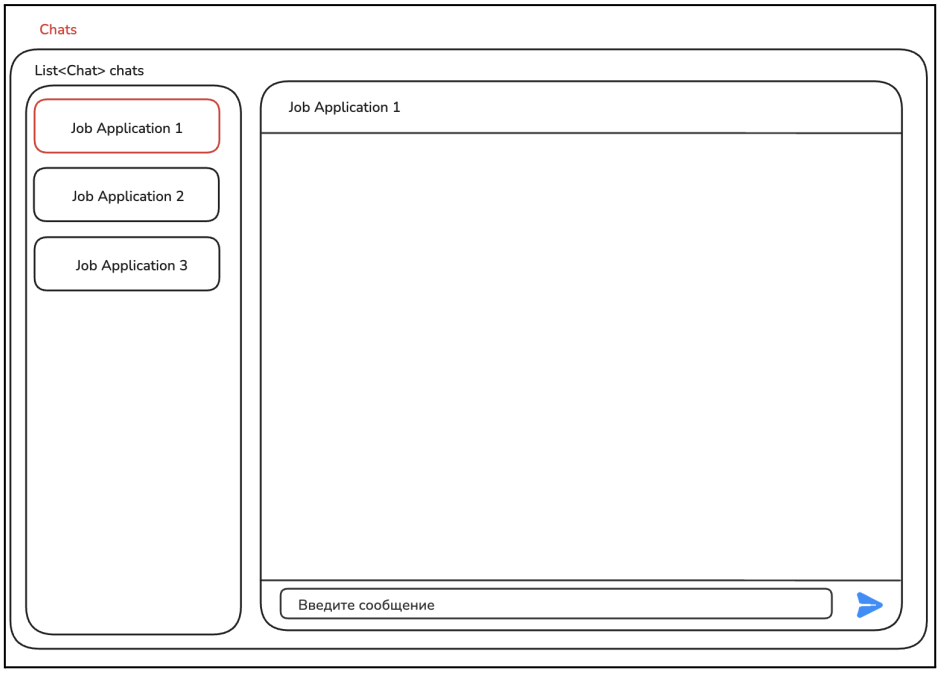


Figure 6. Candidate’s chats list

Recruiter's role:

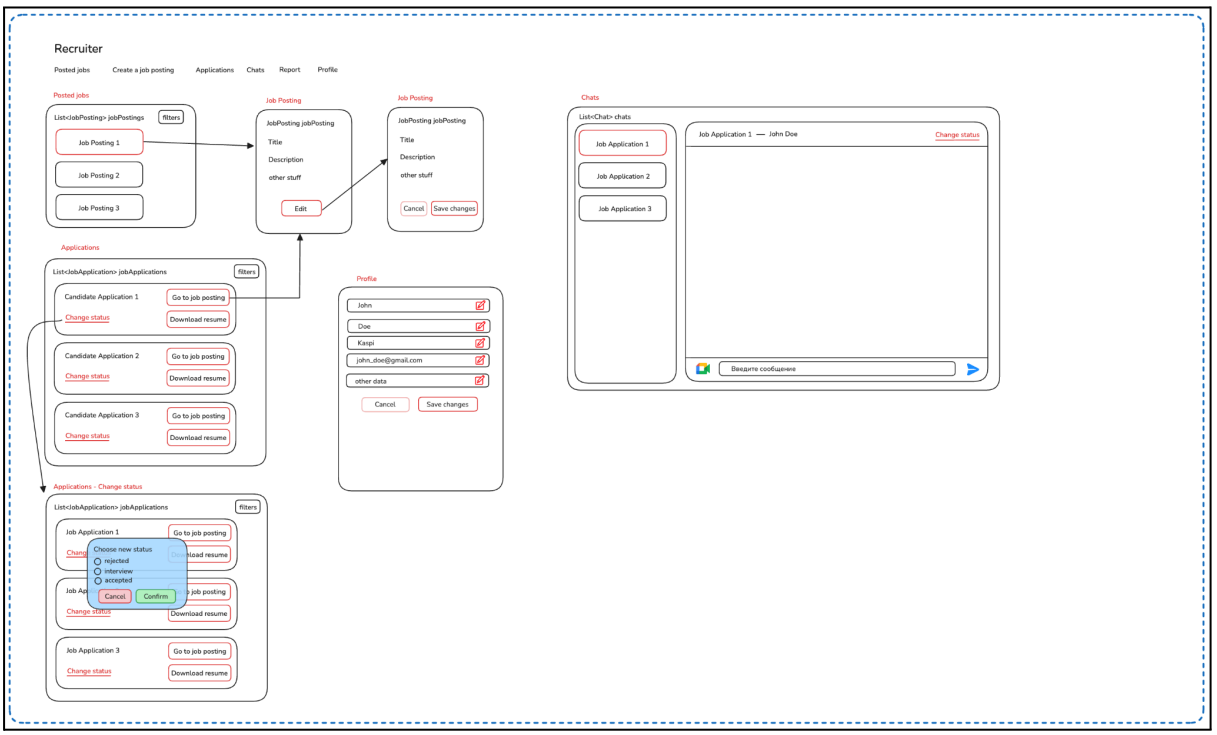


Figure 7. Overview of the whole application from the view of a Recruiter

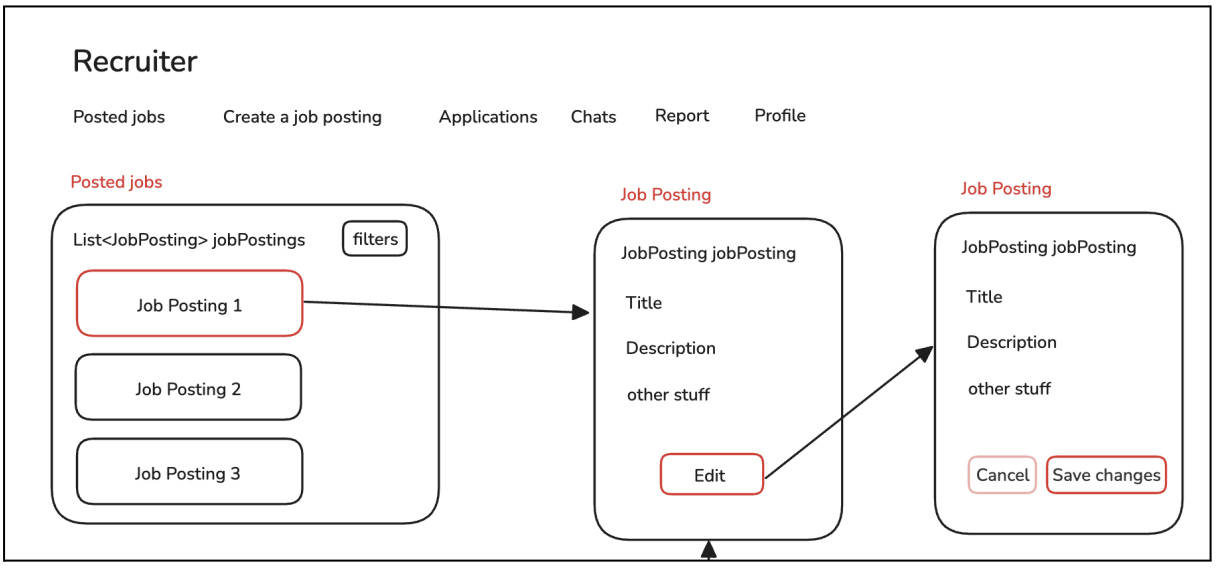


Figure 8. Listing of all posted jobs, a single job page and it's edit dialog

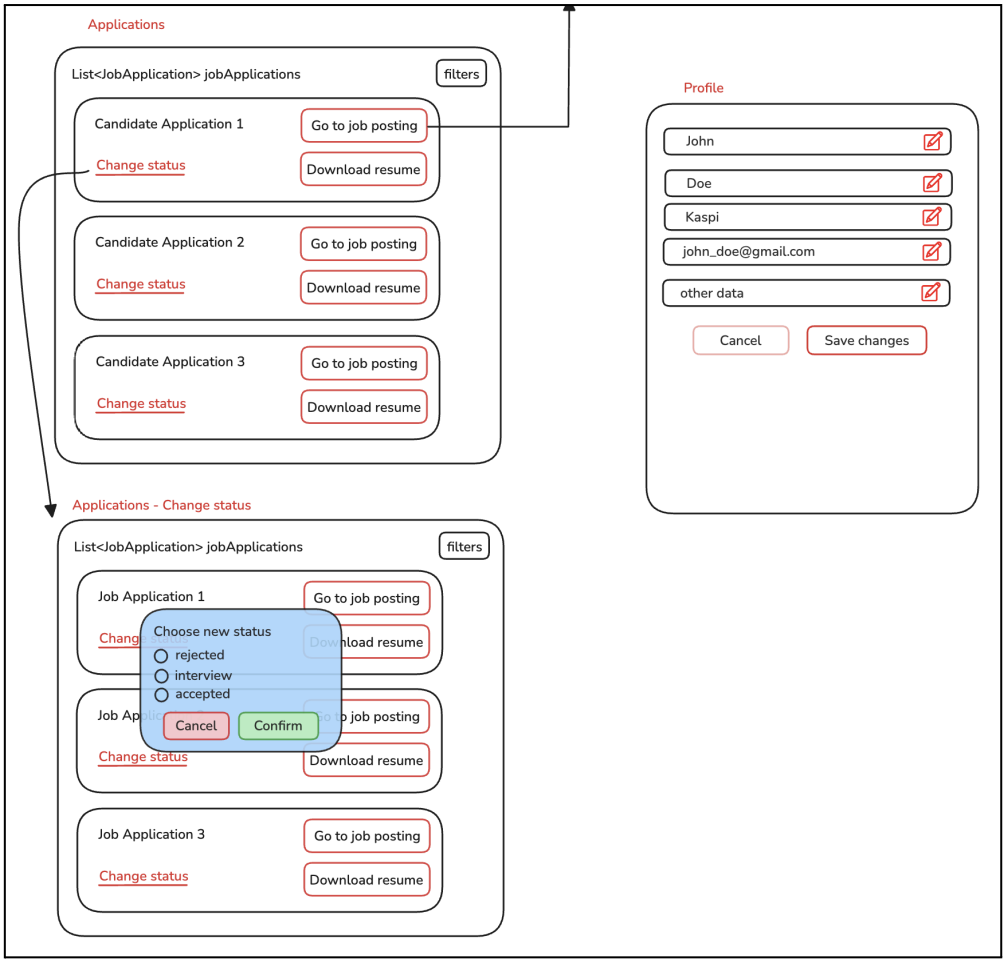


Figure 9. Recruiter can change status of candidates' applications

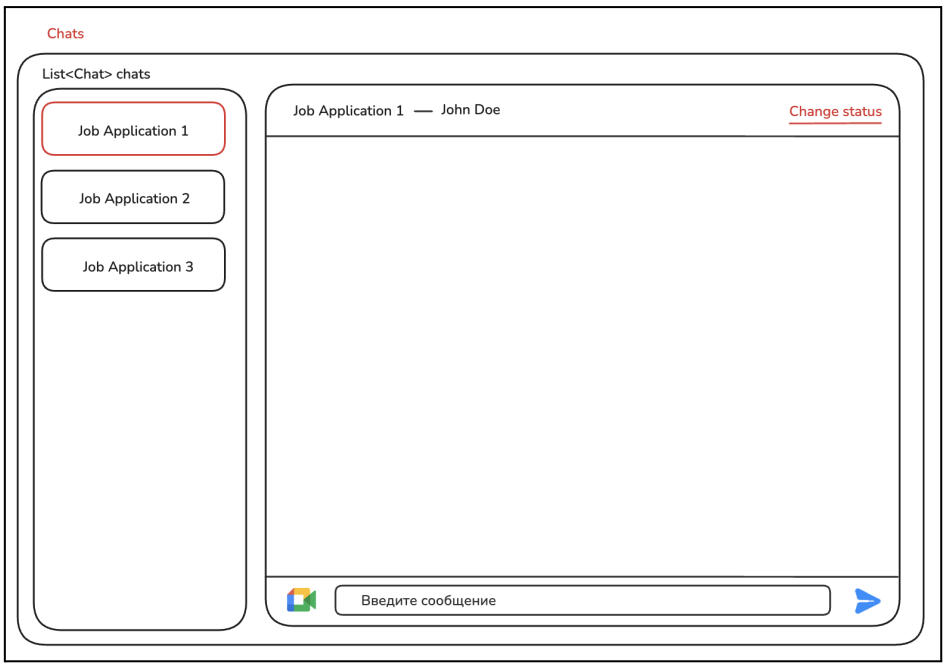


Figure 10. Recruiter's chats list with candidates with the ability to change status, and schedule an online meeting in Google Meet

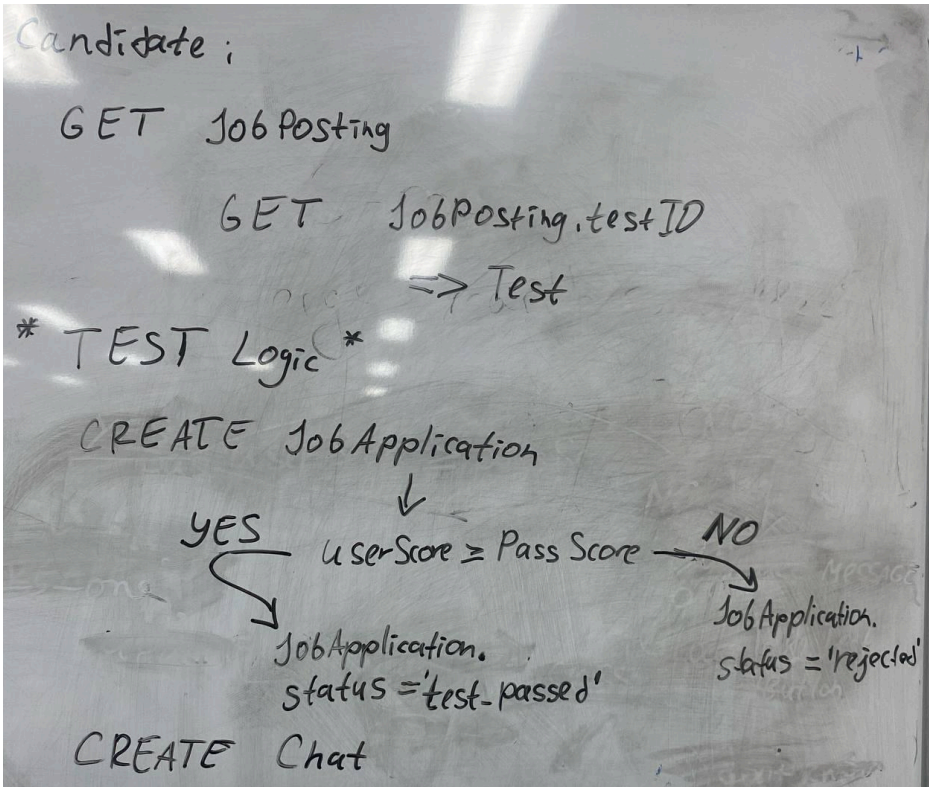


Figure 11. Candidate's perspective on Application creation after the test is passed

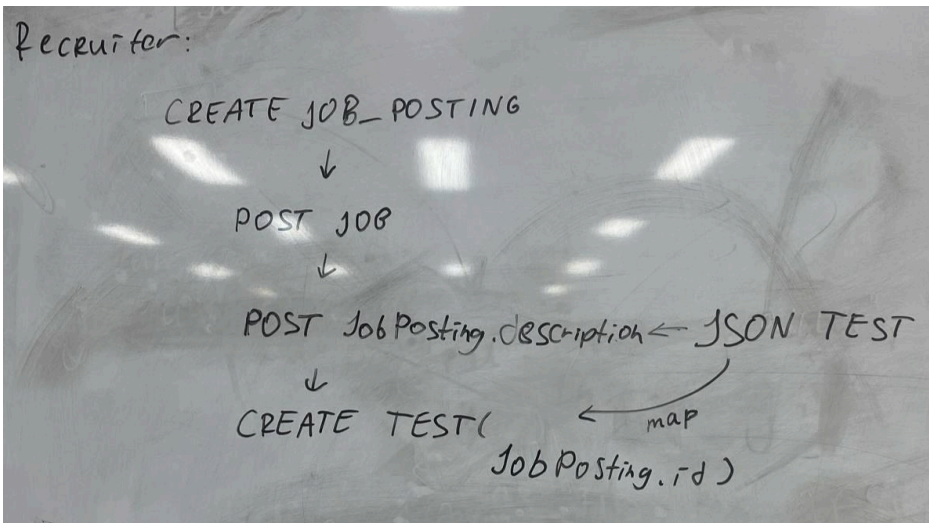


Figure 12. Recruiter's view on integration of AI generated test when the job posting is created

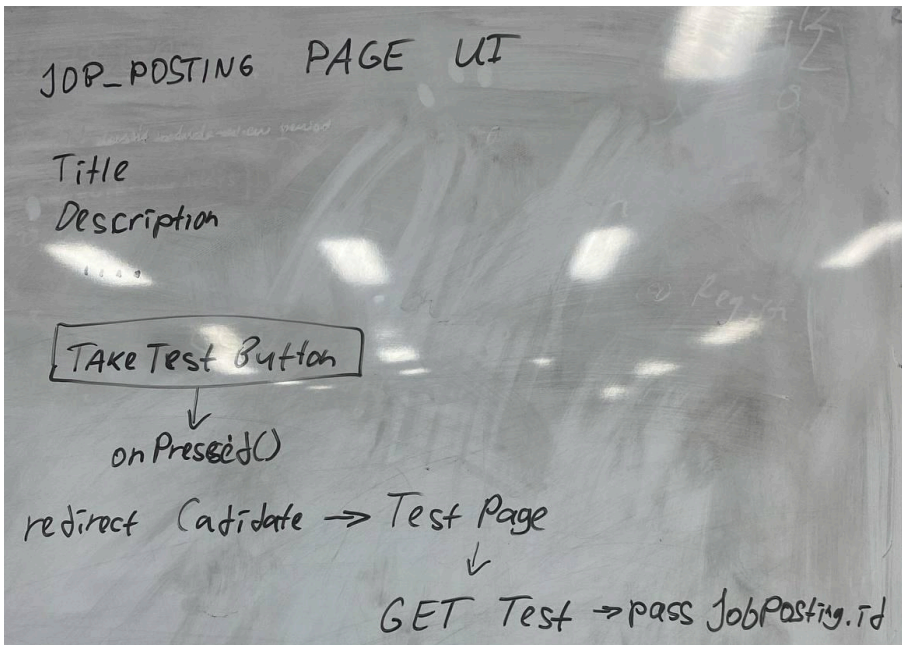


Figure 13. Candidate is on the Job Posting page and can take a test to apply to the vacancy

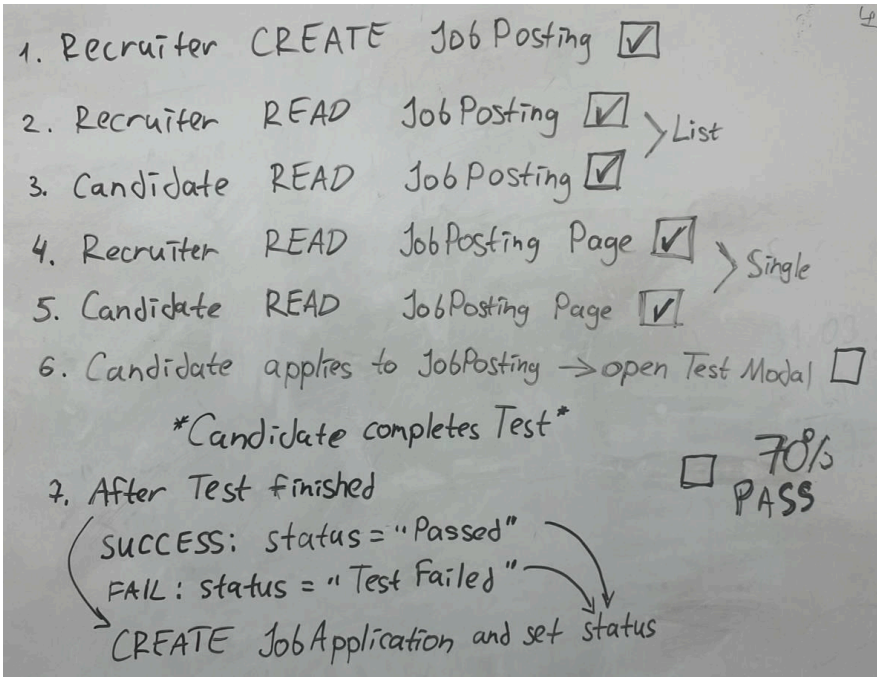


Figure 14. Team's working progress while the project development

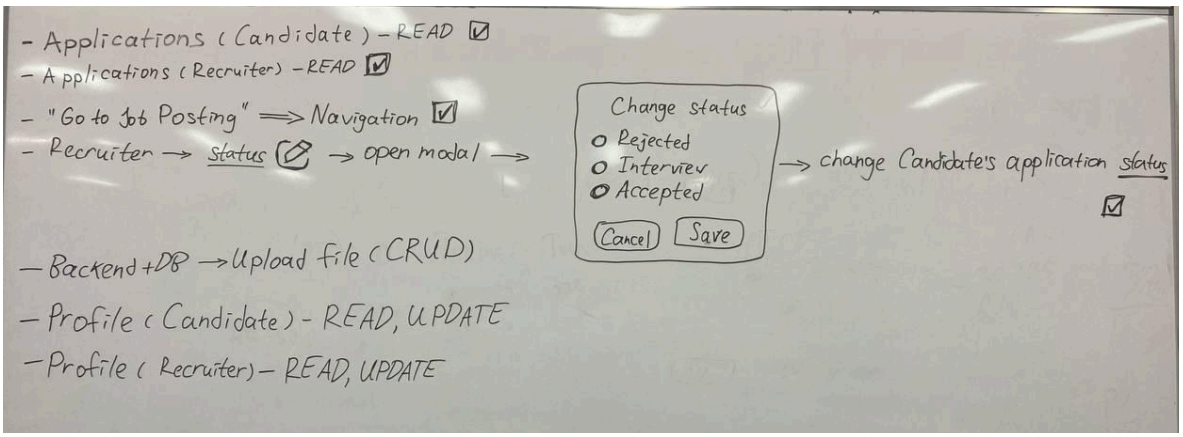


Figure 15. Tasks and features proposed during the development

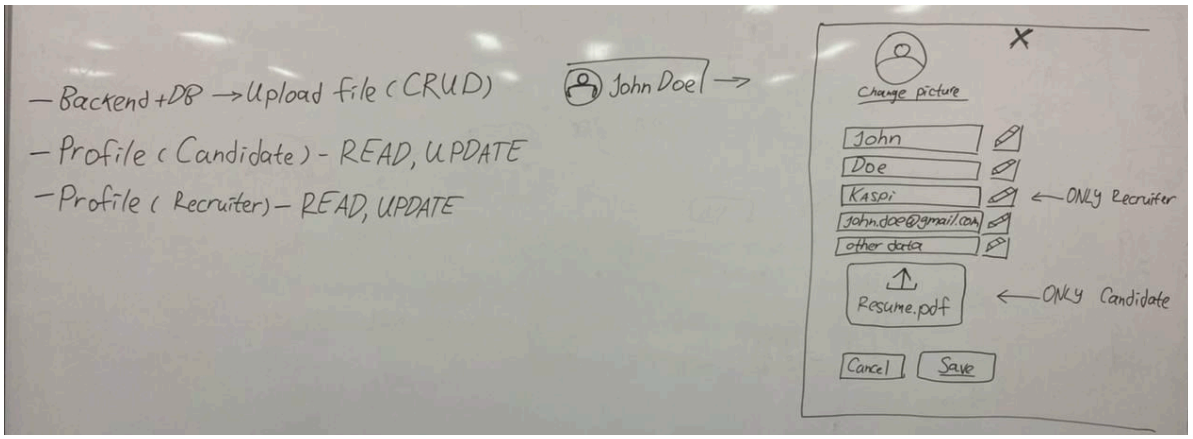


Figure 16. Profile page features and sketch

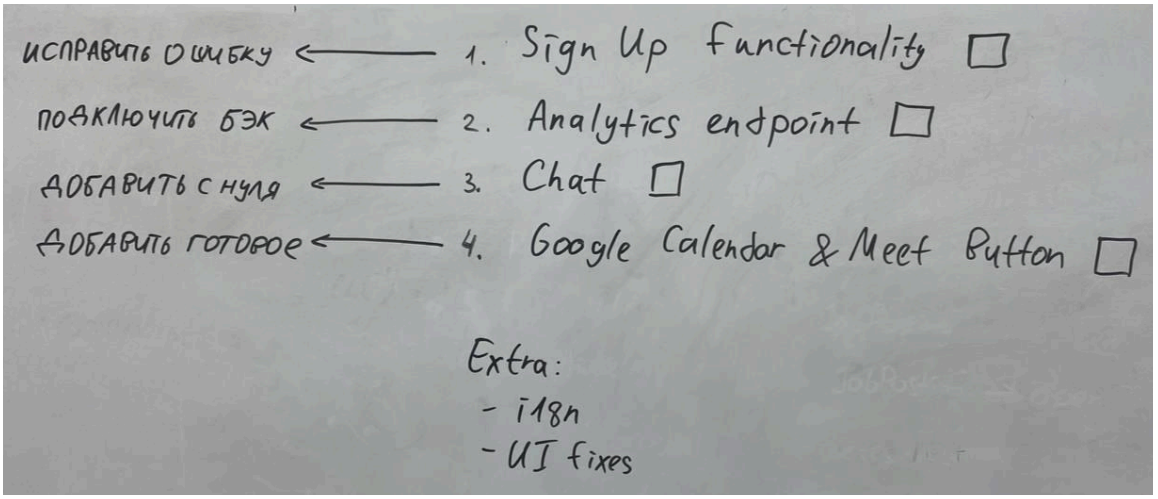


Figure 17. Additional key features proposed list