Job Board API – Technical Report

Author: Bico Steve

Date Created: 06 Dec 2025

Version: [1.0]

Executive Summary

The Job Board API is a backend application designed to streamline job postings, applications, and user account management. It provides secure user registration, Redis-based account verification, JWT-based authentication, and password reset functionality. Administrators can create job openings, manage applications, and update job statuses, while users can build profiles, add education details, browse jobs, and apply for positions. The system is built with Python and Flask-RESTful, backed by MySQL and Redis, containerised with Docker, and integrated with GitHub Actions for automated testing and deployment.

Background and Context

Recruitment is a critical challenge for startups. Outsourcing HR services or relying on third-party job boards can be **expensive, time-consuming, and inefficient**. Startups often need a **focused, cost-effective** recruitment approach that integrates seamlessly into their websites.

The Job Board API solves this problem by:

- Allowing startups to **manage recruitment in-house** without costly HR intermediaries.
- Providing a **modular, extensible backend** that can be embedded into a startup's website.
- Cutting out middlemen and enabling startups to directly connect with candidates.
- Offering a **secure, scalable system** for user registration, verification, job posting, and application management.

This approach empowers startups to control their recruitment pipeline, reduce costs, and maintain a direct relationship with applicants.

User Experience

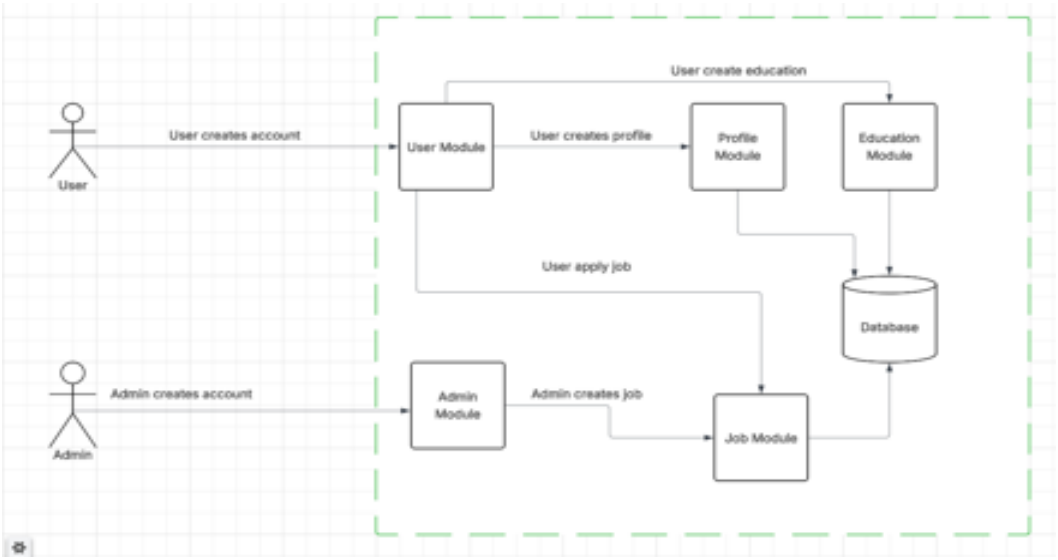
- **Users**: Register accounts, verify via Redis code, log in with JWT, reset passwords, build profiles, add education, browse jobs, and apply.

- **Administrators**: Register admin accounts, create job postings, update job statuses, and view applications.
- **Security**: JWT tokens ensure secure session management, while Redis provides temporary storage for verification and reset codes.

Technical Design

- **Architecture**: Controller → Service → Repository pattern.
- **Modules**:
  - **User Module**: Handles registration, verification, login, and password reset.
  - **Admin Module**: Manages administrator registration and job postings.
  - **Profile Module**: Allows users to build personal profiles.
  - **Education Module**: Enables users to add education details.
  - **Job Module**: Provides job creation, browsing, and application functionality.
  - **Applications Module**: Manages job applications, updates statuses, and lists user applications.
- **Tech Stack**:
  - Python and Flask-RESTful framework.
  - MySQL for persistent storage.
  - Redis for temporary verification and reset codes.
  - Docker for containerisation.
  - GitHub Actions for CI/CD.

System Architecture Diagram



API Endpoints Summary with Role-Based Access

Category	Endpoint Path	Method	Purpose
Health	`/health/check`	GET	Check application health status
User	`/user/register`	POST	Register a new user
	`/user/verify`	POST	Verify user account using Redis code
	`/user/login`	POST	Authenticate the user and issue a JWT token
	`/user/request-reset`	POST	Request password reset (code stored in Redis)
	`/user/reset-password`	POST	Reset user password using verification code
	`/user/me`	GET	Retrieve logged-in user profile
Education	`/education/create`	POST	Create an education record for a user
Profile	`/profile/create`	POST	Create a user profile
	`/profile/get`	GET	Retrieve a user profile

Admin	`/admin/register`	POST	Register a new administrator
	`/admin/login`	POST	Authenticate the admin and issue a JWT token
	`/admin/verify`	POST	Verify the administrator account
	`/admin/jobs/<int:job_id>`	PUT	Modify an existing job posting
Jobs	`/public/jobs`	GET	List all public job postings
	`/public/jobs/<int:job_id>`	GET	Retrieve details of a specific job
	`/admin/jobs/create`	POST	Create a new job posting
Applications	`/applications/job/create`	POST	Submit a job application
	`/applications/job/list`	GET	List all applications for a job
	`/applications/user/list`	GET	List all applications submitted by a user
	`/applications/job/<int:application_id>`	GET	Retrieve details of a specific application
	`/applications/job/update/<int:application_id>`	PUT	Update the status of a job application

Implementation Plan

1. **Setup**: Configure Docker containers for Flask, MySQL, and Redis.
2. **Modules Development**: Implement user, admin, profile, education, job, and applications modules.
3. **Authentication**: Integrate JWT for secure login and session management.
4. **Verification & Reset**: Use Redis for temporary storage of codes.
5. **Testing**: Write unit tests for controllers, services, and repositories.
6. **Deployment**: Automate builds and deployments with GitHub Actions.

Future Considerations

- Add role-based access control for finer-grained permissions.
- Integrate with external job boards or LinkedIn APIs.
- Improve monitoring and observability with tools like Prometheus and Grafana.
- Add rate limiting for authentication endpoints.
- Integrate GitHub Actions to automate deployment to the cloud.
- Integrate email notification to send verification and reset code to users.

Testing Strategy

- **Unit Tests**: Mock DB and Redis connections to test repositories and services.
- **Integration Tests**: Validate end-to-end flows (user registration, job application).
- **CI/CD**: GitHub Actions pipeline runs tests automatically on commits.
- **Error Handling Tests**: Ensure proper exceptions are raised for invalid tokens, DB errors, and schema validation failures.
- **Coverage Goals**: Aim for >80% coverage across modules.

Repository

- **URL**: https://github.com/bicosteve/job-board-api