
# C O M P U T I N G  C O L L E G E

## COMPUTER SCIENCE

## OBJECT ORENTED PROGRAMMING

### Project Documentation


| GROUP MEMBER | Id |
|---|---|
| 1.Bereket S/Mariam | DBU1601439 |
| 2.Eyob Begashaw | DBU1601165 |
| 3.Aster Mamushet | DBU1601401 |
| 4.Bilal Kedir | DBU1701439 |

# Contents of the Documentation

# Job Portal Application Documentation

## Overview

The Job Portal Application is a JavaFX-based platform designed to connect job seekers and employers. It allows job seekers to apply for jobs, save job postings, and manage their profiles, while employers can post job listings, manage applications, and communicate with applicants. The application includes an admin dashboard for user and company management, ensuring secure and controlled access to the platform.

## Features

Below is a detailed explanation of the first five features of the **Job Portal Application** as outlined in the provided documentation. These features cover the core functionalities related to **User Management** and provide a foundation for understanding how users interact with the system.

### 1. Sign Up

Description: The application allows users to register as either Job Seekers or Employers, with appropriate validations to ensure data integrity and security.

Explanation: Users access the sign-up window from the main interface by clicking the "Sign Up" button. They select a role (Job Seeker or Employer) using radio buttons.

Required fields include:

- Full Name: A non-empty string.
- Email: Must match the regex pattern `^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\\.[a-zA-Z]{2,}$` for validity.
- Password: Must be at least 6 characters long.
- Profile Picture (for Job Seekers only): Users must upload an image file (JPG, JPEG, or PNG).

- The system checks if the email is already registered to prevent duplicates.
- Upon successful validation, a new `User` object is created with:
  - User type (Job Seeker or Employer).
  - Status set to "Pending" (awaiting admin approval).
- Profile image path (for Job Seekers) or null (for Employers).
  - The user data is saved to the `job_portal_data.txt` file using serialization.

Key Components:

- showSignUpWindow(Stage): Creates the sign-up UI with role selection and input fields.

- handleSignUp(Stage, String, String, String, String, File): Validates inputs and creates the user.
- saveDataToFile(): Persists user data.

Validation:

- Ensures non-empty name, valid email, sufficient password length, and mandatory profile picture for Job Seekers.
- Displays alerts for invalid inputs using showAlert(String).

Outcome:

- Users receive a confirmation message: "Signed up successfully! Awaiting admin approval."
- The sign-up window closes, and users must wait for admin approval before logging in.

Significance:

- Provides a secure entry point for new users.
- Differentiates between Job Seekers and Employers to tailor subsequent functionalities.
- Ensures data quality through strict validation.

## 2. Login

Description: The application offers a secure login process with validation for email, name, password, and account status.

Explanation:

- Users access the login window by clicking the "Login" button on the main interface.
- Required fields include:
  - Full Name: Must match the registered name.
  - Email: Must match the registered email and the regex pattern `^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\\.[a-zA-Z]{2,}$`.

Password: Must match the registered password.

- Additional features:
- A password visibility toggle (eye icon) allows users to show/hide the password.
- A "Forgot Password?" button redirects to the password reset window.
- The system validates:
- Non-empty inputs for name, email, and password.
- Correct email format.
- Existence of a user with the provided email.

- Matching name and password.

Account status:

- Pending: Users are notified to wait for admin approval.
- Rejected: Users are informed their registration was rejected.
- Blocked: Users are instructed to contact support.
- Approved: Allows successful login.

Upon successful login:

- The `currentUser` is set to the logged-in user.
- Job Seekers are directed to the `showJobSeekerWindow(Stage)` dashboard.
- Employers are directed to either:
- `showTurethEmployerChecker(Stage)` if no company is associated.
- `showCompanyProfileWindow(File, String, String)` if a company profile exists.

Key Components:

- `showLoginWindow(Stage)`: Creates the login UI with input fields and buttons.
- Login button action in `showLoginWindow`: Validates credentials and redirects users.
- `showTurethEmployerChecker(Stage)`: Verifies employer company details.

Validation:

- Prevents unauthorized access by checking user status and credentials.
- Displays alerts for invalid inputs or account issues.

Outcome:

- Approved users access their respective dashboards.
- Unapproved or blocked users receive appropriate error messages.

Significance:

- Ensures secure access to the platform.
- Supports role-based navigation (Job Seeker vs. Employer).
- Integrates with admin-controlled user status for enhanced security.

## 3. Password Reset

Description: Users can reset their password using a generated code sent to their email, ensuring they can regain access if credentials are forgotten.

Explanation:

- Users access the password reset window via the "Forgot Password?" button in the login window.

Required fields:

- Name: Must match the registered name.
- Email: Must match the registered email and the valid email regex.

Steps:

1. Users enter their name and email and click "Send Code."

2. The system verifies the name and email against registered users.

3. If valid, a 6-digit random code is generated using `generateCode()` and displayed (simulated as sent to email).

4. Users enter the code, a new password, and confirm the password.

5. The system checks:

   - Correctness of the entered code.
   - New password length (minimum 6 characters).
   - Matching new and confirm passwords.

6. If valid, the user's password is updated, and data is saved.

   - Alerts are shown for invalid inputs, non-matching credentials, or incorrect codes.

Key Components:

- `showPasswordResetWindow()`: Creates the reset UI with input fields and buttons.
- `generateCode()`: Generates a random 6-digit code.
- `saveDataToFile()`: Updates user data with the new password.

Validation:

- Ensures only registered users can reset passwords.

- Validates code accuracy and password consistency.

Outcome:

- Successful reset displays: "Password reset successfully!"
- The reset window closes, allowing users to log in with the new password.

Significance:

- Provides a recovery mechanism for lost credentials.
- Simulates email-based verification (code displayed for simplicity).
- Maintains security by requiring name and email verification.

## 4. Admin Dashboard

Description: The admin dashboard allows administrators to manage users and companies, including approving, rejecting, blocking, unblocking, or deleting users, and managing Tureth Employer Checker companies.

Explanation:

- Admins access the dashboard by clicking the "Admin" button on the main window and entering the password `job1234` in the `showAdminLoginWindow()`.
- The dashboard (`showAdminDashboard()`) has two sections:

### *1. User Management:*

- A ComboBox displays all users with their name, user type, and status.
- Buttons for actions:

  - Approve: Changes a Pending user's status to Approved.
  - Reject: Changes a Pending user's status to Rejected.
  - Block: Blocks an Approved user if they are inactive (no saved jobs or applications for Job Seekers; no job postings for Employers).
  - Unblock: Restores a Blocked user to Approved.
  - Delete: Permanently removes a user after confirmation.

- Buttons are disabled unless a user is selected and the action is applicable (e.g., can't approve an already Approved user).

### *2. Company Management:*

- A ComboBox lists Tureth Employer Checker companies (name and ID).
- Admins can:

- Add a new company with a unique 9-digit ID and non-empty name.
- Delete a selected company after confirmation.
- All changes are saved using `saveDataToFile()`.

- A "Logout" button returns to the main window.

Key Components:

- `showAdminLoginWindow()`: Authenticates admin access.
- `showAdminDashboard()`: Creates the dashboard UI with user and company management.
- `saveDataToFile()`: Persists changes to users and companies.

Validation:

- Admin password is hardcoded (`job1234`).
- Company IDs must be 9 digits and unique.
- User actions are restricted based on current status.

Outcome:

- Admins can control user access and manage legitimate companies.
- Alerts confirm actions (e.g., "User approved," "Company deleted").

Significance:

- Centralizes user and company oversight.
- Ensures only approved users access the platform.
- Maintains a verified list of companies for employer authentication.

## 5. Job Seeker Features

Description: Job Seekers can view and search job postings, save jobs, apply to jobs, view application history, and manage messages.

Explanation:

- After logging in, Job Seekers access their dashboard via `showJobSeekerWindow(Stage)`.

**Sub-Features:**

***1. View and Search Jobs:***

- A GridPane displays active job postings (not expired) in card format, showing company logo, name, job title, type, location, and deadline.

- A search field filters jobs by title in real-time.
- A ComboBox allows sorting by "Default" or "Company Name (A-Z)."

- Each job card has buttons:

  - View: Displays job details (description, qualifications, etc.) in an alert.
  - Apply: Opens the application form (`showUserApplyWindow(Stage)`).
  - Save: Adds the job to the user's saved jobs list.

### *2. Save Jobs:*

- Clicking "Save" adds the job's unique ID (`title_companyName`) to the user's `savedJobs` list.
- Duplicate saves are prevented with an alert.
- Saved jobs are viewed in `showSavedJobsWindow()`, where users can remove them.

### *3. Apply to Jobs:*

  - Users apply via a default form or a custom form (if defined by the employer).

- Default Form: Includes fields like phone, city, country, LinkedIn, resume, skills, education, and job preferences, with validations (e.g., phone format, mandatory resume).
- Custom Form: Displays employer-specified fields (e.g., CGPA, certifications) with appropriate input types (text or file uploads).
- Successful applications add the user's name to the job's `applicants` list and display a confirmation (`showApplicationSummaryWindow(Stage)`).

### *4. View Application History:*

- `showApplicationHistoryWindow()` lists jobs the user has applied to.
- A "Clear History" button removes the user from all job applicants' lists after confirmation.

### *5. Manage Messages:*

  - `showMessagesWindow()` displays messages where the user is the recipient, with options to reply.
  - Messages include sender, company, job title, content, and company logo.
  - Replies are sent via `showReplyMessageWindow(String, String, String)`.

- The dashboard sidebar shows:

- User profile picture and name.
- Current date.

- Search field and new job notification (green/red based on jobs posted in the last 24 hours).
- Buttons for messages, saved jobs, and application history.
- A "Logout" button returns to the main window.

Key Components:

- `showJobSeekerWindow(Stage)`: Creates the dashboard UI.
- `updateJobGrid(GridPane, List<JobPosting>)`: Populates job cards.
- `createJobCard(JobPosting)`: Builds individual job cards.
- `showUserApplyWindow(Stage)`: Handles application forms.
- `showMessagesWindow()`, `showSavedJobsWindow()`, `showApplicationHistoryWindow()`: Manage respective features.

Validation:

- Job applications enforce mandatory fields and correct formats (e.g., phone, GPA).
- Expired jobs are filtered out automatically.

Outcome:

- Job Seekers can browse, apply, save, and track jobs efficiently.
- Communication with employers is streamlined via messages.

Significance:

- Empowers Job Seekers with comprehensive job search and application tools.
- Enhances user engagement with saved jobs and messaging.
- Supports personalized applications through custom forms.

# Architecture

The application is built using **JavaFX** for the GUI and **Java** for backend logic. It follows a modular design with the following key classes:

# Data Models

- **User**: Stores user information (name, email, password, user type, company name, profile image path, saved jobs, status).
- **Company**: Represents employer companies with name, description, and logo file path.
- **AdminCompany**: Stores predefined companies for Tureth Employer Checker with name and ID.
- **JobPosting**: Contains job details (title, description, job type, vacancies, qualifications, skills, experience, salary range, deadline, location, application method, company name, logo).
- **Message**: Stores communication details (sender, recipient, company name, job title, content, logo file path).
- **ApplicationForm**: Defines custom application forms with company name, job title, and required fields.

# Key Methods

- **start(Stage)**: Initializes the application, loads data, cleans up expired jobs, and displays the main window.
- **showMainWindow(Stage)**: Displays the main window with sign-up, login, and admin buttons.
- **showAdminLoginWindow()**: Authenticates admin access to the dashboard.
- **showAdminDashboard()**: Provides user and company management functionalities.
- **showSignUpWindow(Stage)**: Handles user registration with role selection (Job Seeker or Employer).
- **showLoginWindow(Stage)**: Manages user login with status checks.
- **showTurethEmployerChecker(Stage)**: Verifies employer company details.
- **showJobSeekerWindow(Stage)**: Displays job seeker dashboard with job listings, search, and messaging.
- **showEmployerWindow(Stage)**: Sets up employer profile.
- **showCompanyProfileWindow(File, String, String)**: Displays employer dashboard with job management options.
- **showPostJobWindow()**: Allows employers to post new jobs.
- **showApplicantsWindow()**: Displays applicants for a selected job.
- **showUserApplyWindow(Stage)**: Handles job applications with default or custom forms.
- **loadDataFromFile()** and **saveDataToFile()**: Manage data persistence.

# Usage

1. **Starting the Application**:
    - Run the `main` method to launch the JavaFX application.
    - The main window displays options for signing up, logging in, or accessing the admin dashboard.
2. **Admin Operations**:
    - Login with password `job1234` to access the admin dashboard.
    - Approve/reject/block/unblock/delete users.
    - Add or delete companies in the Tureth Employer Checker.
3. **Job Seeker Workflow**:
    - Sign up with a valid email, name, password, and profile picture.
    - After admin approval, log in to view job listings, apply, save jobs, or manage messages.

    - Complete application forms (default or custom) and upload required files.
4. **Employer Workflow**:
    - Sign up as an employer and verify company details via Tureth Employer Checker.
    - Set up company profile with logo and description.
    - Post jobs, view applicants, create custom application forms, and communicate with job seekers.
5. **Data Management**:
    - Data is automatically saved after actions like user registration, job posting, or message sending.
    - Expired jobs are moved to a trash bin, accessible to employers for restoration or permanent deletion.

# Validation Rules

- **Email**: Must match `^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\\.[a-zA-Z]{2,}$`.
- **Phone**: Must match `^\\+251[97]\\d{8}$` (Ethiopian format).
- **URL**: Must match `^(https?://)?([\\w-]+\\.)+[\\w-]+(/[\\w-./?%&=]*)?$`.
- **GPA**: Must match `^\\d*\\.?\\d{1,2}$` and be between 2.0 and 4.0.
- **Numbers**: Must match `^\\d+$` for fields like experience and vacancies.

# Dependencies

- **JavaFX**: For GUI components and layout.
- **Java AWT**: For file operations and desktop integration (e.g., opening files/links).
- **Java Time**: For handling dates and times (LocalDate, LocalDateTime).
- **Java IO**: For serialization and file handling.

# File Structure

- **job_portal_data.txt**: Stores serialized data for users, companies, job postings, messages, application forms, admin companies, and deleted jobs.
- **Resources**: Contains default images (`default_logo.jpg`, `default_profile.png`, `default_background.png`).

# Running the Application

1. Ensure JavaFX SDK is configured in your IDE or build system.
2. Place default image resources in the appropriate resource directory.
3. Compile and run the `Job` class.
4. Use the admin password `job1234` for admin access.