SOLID STATE PHYSICS LABORATORY

TECHNICAL TRAINING REPORT

MINISTRY OF DEFENSE

Solid State Physics Laboratory

DRDO, Lucknow Road, Delhi

Project

DRDO's JRF/SRF Recruitment Management System

(A MERN-Stack Web Application)

MINISTRY OF DEFENSE

DRDO (Solid State Physics Laboratory), Lucknow Road, Delhi

Internship Report

(29th July, 2024 – 30th December, 2024)

Under the guidance of

Ram Ashish Chouksey

(Scientist 'F', SSPL)

Prepared by

Divisha Bhardwaj

(B. Tech (Computer Science), VIth Year)

Banasthali Vidyapith, Tonk, Rajasthan

CERTIFICATE

Certified that **Divisha Bhardwaj** has carried out the project work titled **DRDO's JRF/SRF Recruitment Management System** from **29th July** to **30th December** for the award of **B. Tech (Computer Science)** from **DRDO(SSPL Lab), Delhi** under my supervision.

The thesis embodies result of original work and studies carried out by the student herself and the contents of the thesis do not form the basis for the award of any other degree to the candidate or to anybody else.

Ram Ashish Chouksey

Designation: Scientist 'F', SSPL

Place:

Date:

Preface

The objective of this project was to develop an efficient and user-friendly **JRF/SRF Recruitment Management System** that streamlines the application process for applicants and administrators. This system was designed to automate the submission, validation, and shortlisting of applications, addressing inefficiencies in the existing manual processes.

Throughout the project, I have put in sincere efforts to accomplish the set objectives within the given time frame. While working on this project, I encountered challenges in integrating multiple technologies such as the MERN stack and MongoDB for real-time functionality. However, with persistent effort, a strong commitment to learning, and the valuable guidance of my supervisor, I successfully navigated these challenges and transformed the objectives into a functional reality.

This project has provided me with a deep understanding of full-stack development and database management systems, enhancing my technical knowledge and problem-solving abilities. It has been a rewarding learning experience, and I am proud to present this carefully crafted effort.

Yours sincerely,

Divisha Bhardwaj

Acknowledgement

I would like to take this opportunity to extend my heartfelt gratitude to all those who guided and supported me throughout this project. Their encouragement and assistance were instrumental in its successful completion.

First and foremost, I would like to express my deepest gratitude to my mentor, **Ram Ashish Chouksey**, for their invaluable guidance and continuous support. Their expertise and insights were pivotal in shaping the direction of this project, and their constructive feedback greatly enhanced the quality of my work.

I am also profoundly grateful to the **team at DRDO (SSPL Lab), Delhi**, who provided critical support and guidance at various stages of the project. Their patience in addressing my queries and their willingness to share knowledge helped me overcome numerous challenges and made this learning experience truly enriching.

Working on the **project** has been a rewarding journey, enabling me to delve into full-stack development and database management. The experience has greatly contributed to my technical growth and problem-solving skills, for which I will always remain thankful.

Lastly, I would like to thank my family and friends for their unwavering encouragement and moral support throughout this endeavour.

Contents

- SOLID STATE PHYSICS LABORATORY (SSPL)
- Vision
- Mission
- About SSPL

❖ **DRDO's JRF/SRF Recruitment Management System**

1. Introduction
2. Objective (Problem Statement)
3. Proposed System
4. Requirement Analysis (SRS)
    a. Requirement Specification
    b. H/w and S/w Requirements
    c. Feasibility Study
    d. Product Functions
    e. Use-case Diagrams
5. System Design (SDS)
    a. High-level Design
    b. ER Diagram/Class Diagrams
    c. Database Design
    d. Data flow diagrams/Activity Diagrams
    e. Flowcharts/Sequence Diagrams
6. Coding
7. Testing
    a. Test cases
8. User Interfaces
9. Appendices
10. References

SOLID STATE PHYSICS LABORATORY (SSPL)

is a premier laboratory under the Defence Research & Development Organization, Ministry of Defence, engaged in the research and development of advanced semiconductor materials and devices.

VISION

To be the centre of excellence in Research and Development of Solid-State Materials, Microelectronics Devices and Nanotechnology for defence applications.

MISSION

Develop semiconductor materials and electronic devices/components for critical defence applications and establish production centres thereof.

Establish scientific knowledge base and build world class scientific research teams/leaders for futuristic cutting-edge technologies.

Strengthen scientific interaction with academia, industry and international centres of excellence.

About SSPL

Solid State Physics Laboratory, a premier DRDO semiconductor research laboratory, was established in 1962 with the broad objective of developing an R&D base in the field of Solid-State Materials, Devices, and sub-systems. Ever since, SSPL has worked relentlessly to achieve distinction in the indigenous development of semiconductor materials and devices.

Several solid-state devices like Gunn, Schottky Barrier and IMPATT Diodes, Monolithic Microwave Integrated Circuits (MMICs), High-power Laser Diodes, Quadrant Detectors, Single Pole Single Throw (SPST) Switches, PIN Photodiodes, space-quality Solar Cells, Accelerometers, Linear Infrared (IR) Arrays, Thermoelectric (TE) Coolers, Phase Shifters, Hydrophones, SAW based e-Nasika etc. have been developed at SSPL. Device quality II-VI/III-V semiconductor crystals and heterostructures have also been developed.

The laboratory is equipped with advanced facilities for Nano-fabrication, Crystal growth and epitaxy, including Electron Beam Lithography, Focused Ion Beam, III-As/N/Sb & II-VI MBEs, III-N/As MOCVDs etc. In addition, an advanced facility for micro-structural, compositional and electrical characterization of semiconductor materials and devices has been setup.

Abstract

The DRDO’s JRF/SRF Recruitment Management System is a web-based application designed to automate the recruitment process for positions like Junior Research Fellows (JRFs) and Research Associates (RAs) at the Solid-State Physics Laboratory (SSPL), DRDO. Traditionally, the recruitment process involved manual submission and evaluation of applications, leading to inefficiencies, inaccuracies, and delays. This project leverages the MERN stack (MongoDB, Express.js, React, and Node.js) to streamline the process, reducing human effort and improving accuracy.

The system enables applicants to fill out multi-step forms online, capturing personal, educational, and professional details along with supporting documents. Real-time validation ensures data accuracy, and submitted data is securely stored in a MongoDB database. Administrators can access submitted applications, apply filters, and shortlist candidates based on predefined criteria such as academic merit, NET/GATE scores, and relevant experience. The system also provides downloadable reports in Word and Excel formats, ensuring compatibility with existing organizational workflows.

Key benefits of the system include enhanced efficiency, reduced manual errors, improved scalability, and a user-friendly interface for both applicants and administrators. By automating the recruitment process, the project aligns with DRDO's mission to foster innovation and operational excellence in its core objectives.

CHAPTER 1

INTRODUCTION

The Defence Research and Development Organisation (DRDO) plays a pivotal role in advancing India's scientific research and defence technologies. The Solid-State Physics Laboratory (SSPL), one of its critical divisions, frequently advertises positions such as Research Associates (RAs) and Junior Research Fellows (JRFs) to attract skilled and talented individuals. Traditionally, these job openings have been publicized through newspaper advertisements, with candidates required to submit physical applications. However, this manual process has proven to be labor-intensive, time-consuming, and prone to errors.

For instance, a single job posting for just two positions may attract up to 1,500 applications, each requiring individual evaluation. Manually reviewing every application, validating credentials, and shortlisting eligible candidates is a resource-draining task, diverting attention from DRDO's core focus on research and development. Additionally, maintaining accurate records for applications and ensuring uniformity in the selection process has posed significant challenges.

To address these inefficiencies, this project introduces the DRDO Recruitment Management System, a web-based solution designed to automate the end-to-end recruitment process. The system enables applicants to seamlessly submit applications via an intuitive React-based frontend interface. Personal, educational, and professional details, along with supporting documents, are securely uploaded and stored in a MongoDB database. Real-time data validation ensures accuracy, while submitted data is exported into Word and Excel formats for compatibility with DRDO's existing workflows.

On the admin side, the system provides a powerful dashboard where administrators can view, filter, and shortlist candidates based on predefined criteria such as academic performance, NET/GATE scores, and relevant experience. Filters can be applied dynamically to refine the applicant pool, and shortlist statuses are updated directly in the database. This eliminates the need for manual evaluation, significantly reducing human effort and errors. Moreover, administrators can generate comprehensive reports for further analysis or archival purposes.

By automating the application submission, validation, and shortlisting processes, the DRDO Recruitment Management System enhances efficiency, accuracy, and scalability. This solution allows DRDO SSPL to allocate its

resources more effectively toward its primary objective of fostering innovation and technological advancement.

CHAPTER 2

PROBLEM STATEMENT

The DRDO's Solid-State Physics Laboratory (SSPL) is pivotal in enhancing India's defense technology by recruiting exceptional talent for positions such as Junior Research Fellows (JRFs) and Research Associates (RAs). The existing recruitment process predominantly depends on manual procedures and conventional methods, which pose several challenges hindering efficiency and accuracy.

Overview of the Current Process:

1. **Advertisement:**

    - Job vacancies are disseminated through newspapers and various public mediums to attract qualified applicants.
    - Applicants are required to submit paper-based applications, typically via mail to the DRDO office.

2. **Application Review:**

    - Each application undergoes a manual evaluation to determine the candidate's qualifications, academic history, and credentials.
    - The procedure requires the evaluation of application details against job-specific criteria, including NET/GATE scores, academic qualifications, and relevant experience.

3. **Shortlisting:**

    - From a large pool of applicants (e.g., 1,500 for two vacancies), only a small percentage (typically about 60 candidates) are shortlisted for interviews based on predetermined criteria.

4. **Data Management:**

- Candidate information is maintained in physical records, resulting in cumbersome retrieval and reporting.
- The absence of a structured, centralized system complicates the compilation of shortlists and reports.

Major Obstacles:

1. **High Volume of Applications:**

   - A single job advertisement can attract hundreds or even thousands of applications, overwhelming the manual evaluation process.
   - Managing this volume necessitates considerable human effort, frequently resulting in delays in the shortlisting process.

2. **Time-Intensive and Inefficient:**

   - The manual procedure of verifying applications, assessing criteria, and selecting candidates requires considerable time investment.
   - Administrative staff must allocate considerable resources to tasks that could be automated.

3. **Error-Prone and Inconsistent:**

   - The heavy dependence on personal involvement enhances the likelihood of human errors, inconsistencies, and biases.
   - Competent candidates may be overlooked due to oversight, whereas less suitable candidates may be unintentionally chosen.

4. **Data Retrieval and Reporting Issues:**

   - Physical documents and unstructured data make information retrieval tedious and ineffective.
   - The generation of structured reports or shortlists requires considerable manual labor, resulting in operational delays.

Implications of the Existing System:

- **Reduced Productivity:** The allocation of time and resources to repetitive manual tasks diminishes the organization's capacity to concentrate on fundamental research and innovation.
- **Inadequate Recruitment Decisions:** Mistakes or delays in the recruitment process can lead to the loss of opportunities to recruit top talent for critical roles.
- **Operational Inefficiency:** The management of huge data volumes without automation results in inefficiencies in data processing and decision-making.

Need for an Automated System:

To address these challenges, there is a clear need for an automated and efficient recruitment process that utilizes modern technologies to:

- Simplify application collection and processing.
- Minimize human intervention to decrease errors and biases
- Enhance data management via centralized storage and retrieval systems.
- Accelerate shortlisting and reporting, ensuring faster decision-making and hiring.

An automated recruitment management system will significantly improve the efficiency, accuracy, and scalability of SSPL's hiring process, allowing the organization to attract and assess talent more effectively. This modernization corresponds with DRDO's objective to utilize advanced technologies in its operations.

CHAPTER 3

PROPOSED SYSTEM

The **DRDO's JRF/SRF Recruitment Management System** is designed to modernize and streamline the entire recruitment process for positions like Junior Research Fellows (JRFs) and Research Associates (RAs). By addressing the inefficiencies of manual application handling, the proposed system automates key aspects such as data collection, validation, shortlisting, and reporting. Built using the MERN stack—MongoDB, Express.js, React, and Node.js—the system ensures scalability, accuracy, and an enhanced user experience.

The system consists of two primary interfaces:

- Applicant Interface: Enables candidates to fill and submit forms.
- Admin Interface: Allows administrators to manage applications, apply filters, and generate reports.

Key Features of the Proposed System

1. Frontend: React

The frontend is built using React, offering a dynamic and user-friendly experience for applicants.

- **Multi-Step Form:**
  - Divides the application into logical steps: Personal Details, Educational Qualifications, Work Experience, Document Upload, and Review & Submit.
  - Progress indicators help users track their position in the application process.
- **Real-Time Validation:**
  - Validates inputs like name (in block letters), email, phone numbers, and file types before submission.
  - Ensures mandatory fields are filled while allowing flexibility for optional sections.
- **Responsive Design:**
  - Accessible on desktops, tablets, and smartphones, catering to a wide range of users.

2. Backend: Node.js and Express.js
The backend ensures smooth communication between the frontend and database through secure APIs.

- **Secure APIs:**
  - Facilitates form submissions, data retrieval, and file uploads.
- **Data Validation:**
  - Uses middleware for sanitizing inputs and validating the integrity of submitted data.
- **File Uploads:**
  - Handles the upload of photos and documents securely, ensuring only acceptable file formats (JPEG, PNG, PDF) are processed.

3. Database: MongoDB
MongoDB serves as the backbone of the system's data storage, offering flexibility and performance.

- **JSON-Like Documents:**
  - Stores candidate data, including personal information, qualifications, and document paths, in a structured format.
- **Scalability:**
  - Can handle increasing volumes of applications without compromising retrieval or write speeds.
- **Schema Enforcement:**
  - Ensures consistency across records, enabling efficient query operations.

4. Admin Panel
The admin panel is designed to empower administrators by providing robust features for application management.

- **Application Overview:**
  - Displays a list of all submitted applications with key details such as name, discipline, and application status.
- **Filtering and Search:**
  - Advanced filters allow admins to shortlist candidates based on academic merit, NET/GATE scores, and relevant experience.
  - Search functionality enables quick retrieval of specific applications.

- **Shortlisting:**
  - Admins can mark candidates as shortlisted or rejected and update the database in real time.

- **Reporting:**
    - Generates Excel and Word documents containing organized candidate data for review and archival purposes.

Workflow of the Proposed System

1. **For Applicants:**
    - Register or log in to access the application form.
    - Fill out personal, educational, and professional details in a step-by-step format.
    - Upload supporting documents such as photos, marksheets, and declarations.
    - Review and submit the application, receiving a confirmation upon successful submission.
2. **For Admins:**
    - Access the admin dashboard to view all applications.
    - Apply filters and search criteria to shortlist candidates dynamically.
    - Export candidate data into easily shareable formats like Excel or Word for further processing.

Advanced Features

- **Real-Time Updates:**
    - Changes made to applications or shortlists are reflected instantly, ensuring up-to-date records.
- **Adaptable Shortlisting Algorithm:**
    - Considers predefined criteria like academic performance, NET/GATE scores, and work experience.
    - Can be customized to match evolving recruitment needs.
- **Error Handling:**
    - Provides detailed error messages for both applicants and admins, ensuring smooth operation.

Benefits of the Proposed System

- **Efficiency:** Automates repetitive tasks like data entry, shortlisting, and reporting, saving significant time.
- **Accuracy:** Reduces human errors through validation and automation.
- **Scalability:** Handles large volumes of applications effortlessly.
- **Security:** Protects sensitive data through secure APIs, role-based access control, and encrypted file storage.
- **User-Friendly**: Enhances the experience for both applicants and admins with intuitive design and real-time feedback.

CHAPTER 4

REQUIREMENT ANALYSIS

4.A. REQUIREMENT SPECIFICATIONS

1. Functional Requirements

The functional requirements describe the core features and operations of the DRDO Recruitment Management System:

1. **User Registration and Authentication**:
    - Secure registration for new applicants.
    - Login functionality for returning applicants to access and complete forms.
    - Role-based access for administrators to manage applications and reports.
2. **Application Form Submission**:
    - Multi-step form for candidates to submit:
        - **Personal Details**: Name, contact information, gender, date of birth.
        - **Educational Background**: Degree, board, passing year, percentage, and uploaded marksheets.
        - **Professional Experience**: Positions held, organizations, duration, and responsibilities.
        - **Additional Credentials**: NET/GATE qualifications and year of pasSLing.
    - Real-time validation of form fields to ensure correct data entry.
    - Support for document uploads (photos, certificates, publications, PhD-related files).
3. **Data Management**:
    - Use of a **MongoDB database** to store and manage candidate information securely.
    - Capability to retrieve, update, and back up candidate data efficiently.
    - Export options for candidate information in **Word** and **Excel** formats.
4. **Automated Shortlisting**:
    - Algorithm to automatically shortlist candidates based on:

        - Academic qualifications.
        - NET/GATE scores.
        - Relevant work experience.
    - Dynamic filtering options for admins to adjust shortlisting criteria.
5. **Report Generation**:
    - Generate reports summarizing:
        - Candidate details.
        - Shortlisting results.
    - Allow admins to download reports in multiple formats for further analysis.
6. **Admin Panel**:
    - Interface for administrators to:
        - View and manage submitted applications.
        - Apply filters dynamically for shortlisting.
        - Generate structured reports.

2. Non-Functional Requirements

These requirements ensure the system's overall quality and reliability:

1. **Performance**:
    - Handle simultaneous access by multiple users without noticeable delays.
    - Optimize response times for form submissions, data retrieval, and report generation.
2. **Security**:
    - Encrypt data during storage and transmission.
    - Implement role-based access control to secure sensitive data.
    - Enforce strong password policies and support multi-factor authentication for admin users.
3. **Scalability**:
    - Design the system to handle increasing numbers of users and applications.

- Support horizontal scaling for managing heavy traffic and data loads.

4. **Usability**:
    - User-friendly interface with clear instructions and multi-step navigation.
    - Mobile-friendly design for compatibility across devices.
5. **Maintainability**:
    - Modular code structure for easier debugging and feature addition.
    - Well-documented system components to ensure smooth updates.
6. **Reliability**:
    - Ensure data integrity through validations and error handling.
    - Guarantee accurate and consistent functionality for all operations.

3. Software Quality Attributes

To enhance the user experience and maintain system quality:

1. **Availability**:
    - Ensure 24/7 uptime with minimal maintenance downtime.
    - Handle sudden spikes in user activity efficiently.
2. **Portability**:
    - Compatibility with all major browsers and platforms for uniform functionality.
3. **Reusability**:
    - Modular design for reusing components like form validation, shortlisting algorithms, and user authentication in future projects.
4. **Robustness**:
    - Handle unexpected inputs gracefully to avoid crashes.
    - Provide clear error messages and recovery options for users.

4.B. H/W & S/W REQUIREMENTS

Hardware Requirements

Development Environment

- **CPU**: Quad-core processor or higher for local development and testing.
- **RAM**: Minimum **8 GB** to run:
    - IDEs like **Visual Studio Code**.
    - Local **MongoDB** server.
    - **React development server**.
- **Storage**: SSD with at least **50 GB** of free space for:
    - Project files.
    - Node modules.
    - Databases.
- **Operating System**: Compatible with:
    - **Windows 10** or above.
    - **Ubuntu 20.04+** (Linux).
    - **macOS Mojave** or higher.

Deployment Server

- **System Type**: **64-bit architecture** with x64 CPU.
- **RAM**: Minimum **8 GB**, recommended **16 GB** for high user traffic.
- **Storage**: **SSD storage** for fast read/write operations on logs and uploaded files (PDF documents, images, etc.).
- **Operating System**:
    - Preferred: **Linux-based servers** (e.g., Ubuntu 20.04+).
    - Optional: **Windows Server** for enterprise use.
- **Network**:
    - **Gigabit Ethernet** or high-speed **Wi-Fi** for reliable data communication.

Client Devices

- **Devices**:
    - Desktop, Laptop, Tablet, or Smartphone.
- **Browser Support**:
    - Modern browsers such as **Chrome**, **Firefox**, **Edge**, and **Safari**.
- **Operating System Compatibility**:
    - **Windows**, **macOS**, **Linux**, **iOS**, and **Android**.

Software Requirements

Development Side

- **Frontend Development**:
    - **React.js**: For creating responsive and dynamic user interfaces.
    - **Node.js**: For running the development server.
    - **CSS**: For styling components.
    - **Axios**: For making HTTP requests to the backend.
- **Backend Development**:
    - **Express.js**: Web framework for creating APIs.
    - **Multer**: Middleware for handling file uploads.
    - **Joi**: For input validation and schema enforcement.
- **Database**:
    - **MongoDB**: NoSQL database for storing:
        - Candidate details.
        - Uploaded documents.
        - Application statuses.
- **Development Tools**:
    - **Code Editor**: **Visual Studio Code** or similar IDE.
    - **Version Control**: **Git** for managing code repositories.

Deployment Side

- **Web Server**:
    - **Nginx** or **Apache** for serving:
        - React frontend application.
        - Backend APIs.
- **Backend**:
    - **Node.js runtime environment** with all required dependencies.
- **Database Server**:
    - **MongoDB** with appropriate indexing for efficient data queries.

Security Tools:

- **SSL/TLS Certificates**: To secure communication between clients and the server.
- **Environment Variables**:
    - Managed using **.env files** to protect sensitive data such as:
        - Database URIs.
        - API keys.

4.C. FEASIBILITY STUDY

1. Technical Feasibility

- The system is built using the **MERN stack**:
    - **MongoDB**: Manages large-scale candidate data, including personal information, educational qualifications, and document uploads.
    - **Express.js**: Handles backend APIs for operations like form submission, validation, and data retrieval.
    - **React.js**: Provides a responsive and interactive user interface for applicants and administrators.
    - **Node.js**: Ensures seamless server-side execution and integrates efficiently with the database.
- **Key Features**:
    - Multi-step form for applicants with file upload capability (handled by **Multer**).
    - Role-based access for users (applicants) and admins (backend services).
    - Real-time validation and dynamic filtering in the admin panel.
- **Infrastructure**:
    - The project requires moderate hardware specifications (e.g., multi-core CPU, 8 GB RAM, SSD storage) for local development and deployment.
    - Well-documented APIs and modular codebase make maintenance and scaling straightforward.
- **Conclusion**: The choice of stack and tools ensures high scalability, maintainability, and ease of implementation, making the project **technically feasible**.

2. Operational Feasibility

- The system effectively automates **manual tasks**:
    - **Form submissions**: Replaces paper-based processes with a digital platform for accurate and efficient data collection.
    - **Shortlisting**: Automates the filtering of candidates based on predefined criteria (e.g., NET/GATE scores, work experience).

    - **Data management**: Stores all candidate information securely in a **MongoDB database**.

- **Admin Panel**:
    - Features dynamic filters to adjust shortlisting parameters in real time.
    - Provides structured reports for further analysis, saving time and effort.
- **Ease of Use**:
    - The React-based interface is intuitive, with clear navigation for both applicants and admins.
    - Real-time error feedback reduces user mistakes during form submissions.
- **Conclusion**: By minimizing manual intervention and improving workflow efficiency, the project is highly **operationally feasible**.

3. Economic Feasibility

- The system's development uses open-source technologies (**MongoDB**, **Node.js**, **React.js**), significantly reducing software costs.
- **Initial Costs**:
    - Infrastructure setup: Development machines and deployment server.
    - Development team effort.
- **Savings**:
    - Reduces administrative workload by automating repetitive tasks like form evaluation and candidate shortlisting.
    - Minimizes human errors, reducing costs associated with incorrect candidate selections.
- **Long-term Benefits**:
    - Increased recruitment efficiency.
    - Faster decision-making leads to timely hiring, indirectly saving operational costs.
- **Conclusion**: With reduced long-term administrative overhead and minimal initial investment, the system is **economically feasible**.

4. Legal Feasibility

- The system ensures **data privacy and security**:
    - Uses **encryption** to protect candidate data both in transit and at rest.
    - Implements **role-based access control** to prevent unauthorized access.
    - Collects user consent during registration and form submission to comply with data protection laws (e.g., GDPR or local equivalents).
- **Conclusion**: With robust data security measures and adherence to privacy laws, the project is **legally feasible**.

5. Schedule Feasibility

- The project follows a structured timeline with clear milestones:
    - **Frontend Development**: Multi-step form and admin dashboard using React.js.
    - **Backend Development**: APIs and database integration with Node.js and MongoDB.
    - **Testing**: Extensive testing for form validation, shortlisting algorithms, and admin panel functionalities.
    - **Deployment**: Hosted on a secure server with performance optimizations.
- The modular structure of the code allows for parallel development of features, reducing overall development time.
- Contingency plans address potential delays in development or testing.
- **Conclusion**: Given the well-defined timeline and a competent team, the project is **schedule feasible**.

4.D. PRODUCT FUNCTIONS

1. Application Form Submission

- **Multi-Step Form**:
    - **Personal Information**:
        - Includes name, gender, date of birth, contact information, and email.
    - **Academic Details**:
        - Degree, board, year of passing, percentage, and file upload for marksheets.
        - Allows multiple academic entries.
    - **Work Experience**:
        - Includes fields for position, company, duration, and responsibilities.
        - Supports multiple entries for professional experience.
    - **Additional Details**:
        - PhD-related information, publications, and legal declarations.
        - File uploads for relevant documents, including passport-sized photos.

2. Admin Dashboard

- **Application Management**:
    - View all submitted applications in a tabular format.
    - Includes easy access to uploaded documents for review.
- **Dynamic Filtering**:
    - Filters applications based on:
        - Academic scores.
        - NET/GATE qualifications.
        - Duration of work experience.
- **Status Updates**:
    - Admins can update candidate status to shortlisted, rejected, or under review.

3. File Management

- **Document Handling**:
    - Supports uploads for documents such as:
        - Academic certificates.
        - Photos.
        - Self-declaration files.
    - Validates file formats (PDF, JPEG/PNG) and size limits.
- **Secure Storage**:
    - Organizes uploaded files on the server in a structured directory.

4. Real-Time Validation

- **Frontend Validation**:
    - Validates user inputs for correct formats and mandatory fields.
    - Examples:
        - Email format validation.
        - Numeric constraints for year and percentage.
        - File type checks for uploaded documents.
- **Backend Validation**:
    - Ensures complete and accurate data submission before processing.

5. Scalability and Usability

- **Responsive Design**:
    - The system is accessible and optimized for devices including desktops, laptops, and smartphones.
- **Simplified Navigation**:
    - The multi-step application form makes data entry straightforward.
    - The admin panel allows efficient application management.

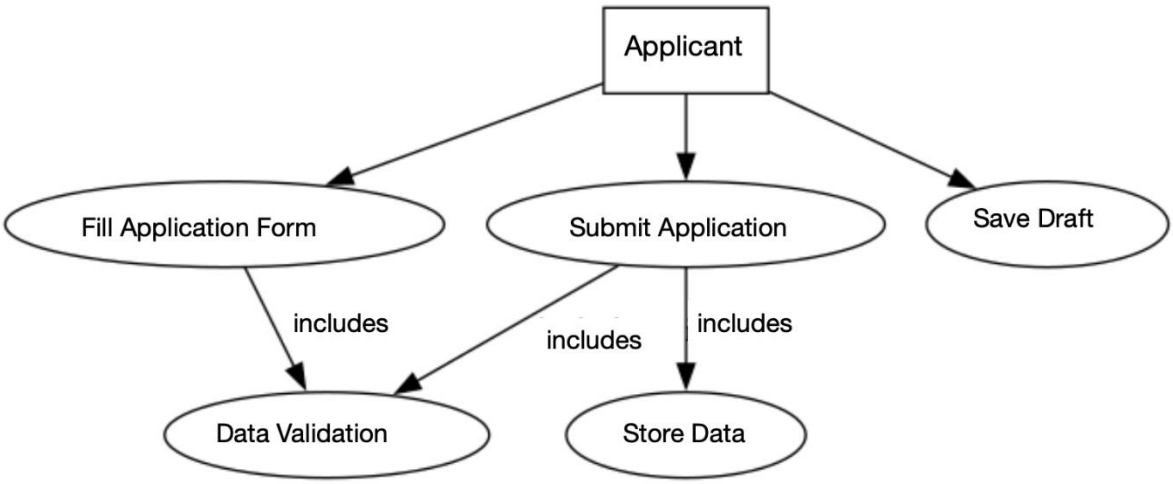
6. User Experience Enhancements

- **Progress Tracking**:
    - Displays a progress bar or step indicator in the multi-step application form to help users understand their current stage in the process.
- **Draft Saving**:
    - Allows users to save partially filled forms and resume later without losing their data.
- **Error Feedback**:
    - Highlights specific fields with errors and provides clear guidance for corrections (e.g., "Enter a valid percentage between 0 and 100").

4.E. USE CASE DIAGRAMS

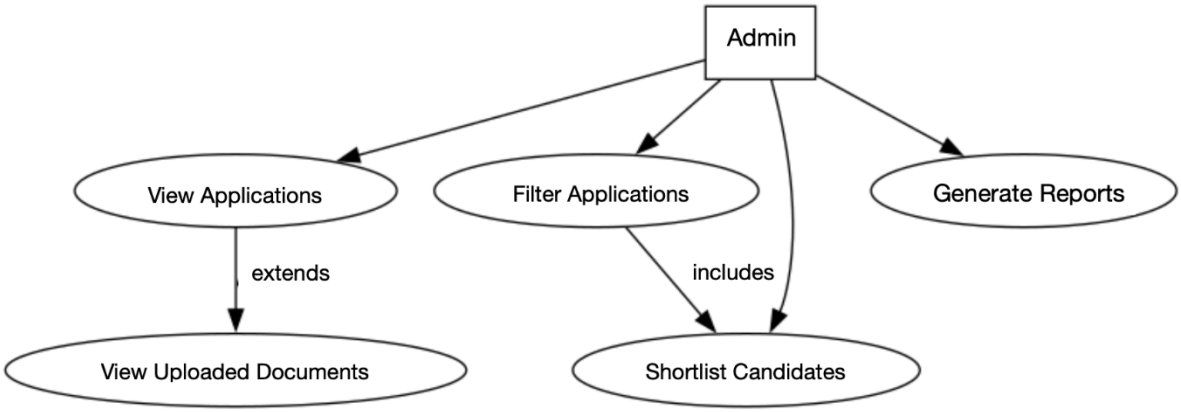
- **(Applicant)**

This **use case diagram** illustrates the actions performed by an **Applicant**, including filling out an application form, saving drafts, and submitting applications, with **data validation** and **data storage** as key dependencies.



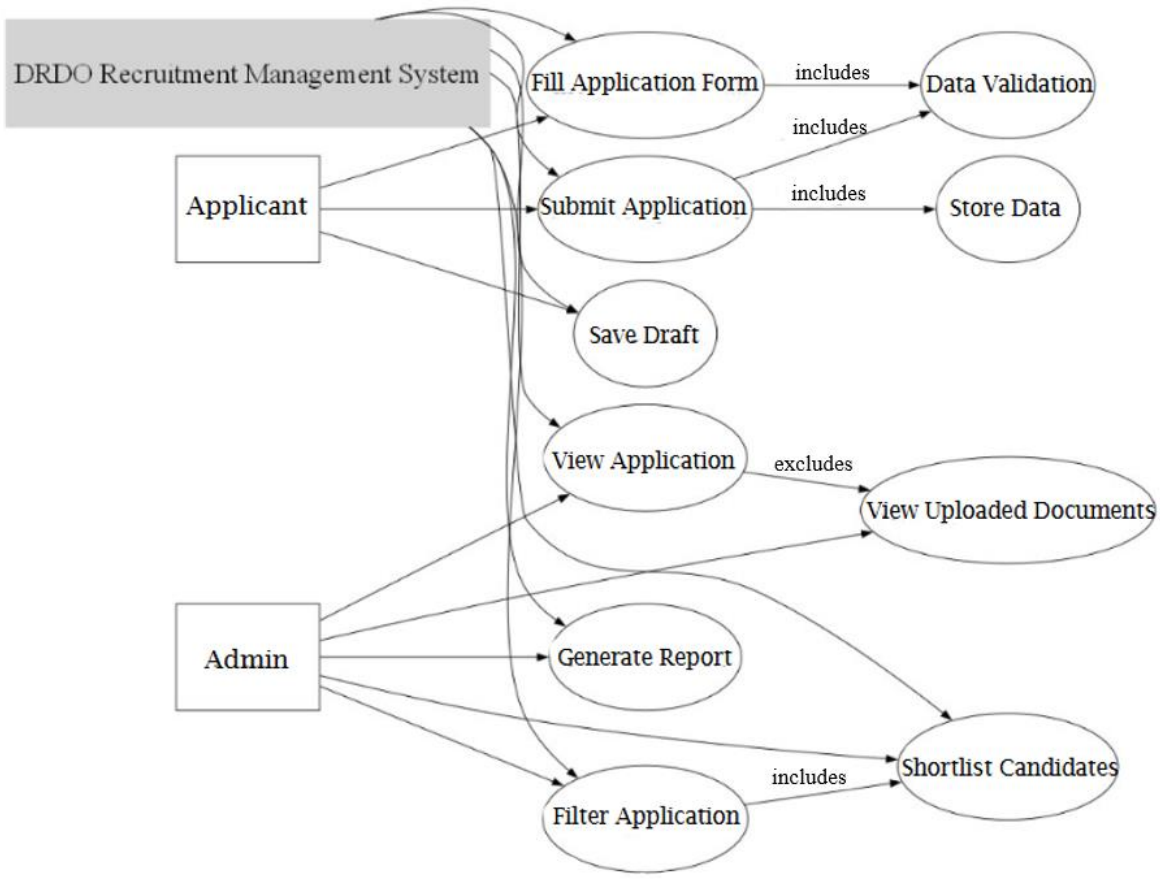
- **(Admin)**

This **admin use case diagram** highlights the actions performed by the **Admin**, including **viewing applications**, filtering them to **shortlist candidates**, generating reports, and extending functionality to **view uploaded documents** for detailed review.



- **(Comprehensive of the whole System)**

This use case diagram for the DRDO Recruitment Management System outlines the interactions between two primary actors: Applicant and Admin. Applicants can perform tasks like filling out the application form, saving drafts, and submitting applications, which include data validation and storing data. Admins manage applications through tasks like filtering applications, shortlisting candidates, and generating reports. Relationships like includes and excludes showcase dependencies and restrictions within the system's functionalities.



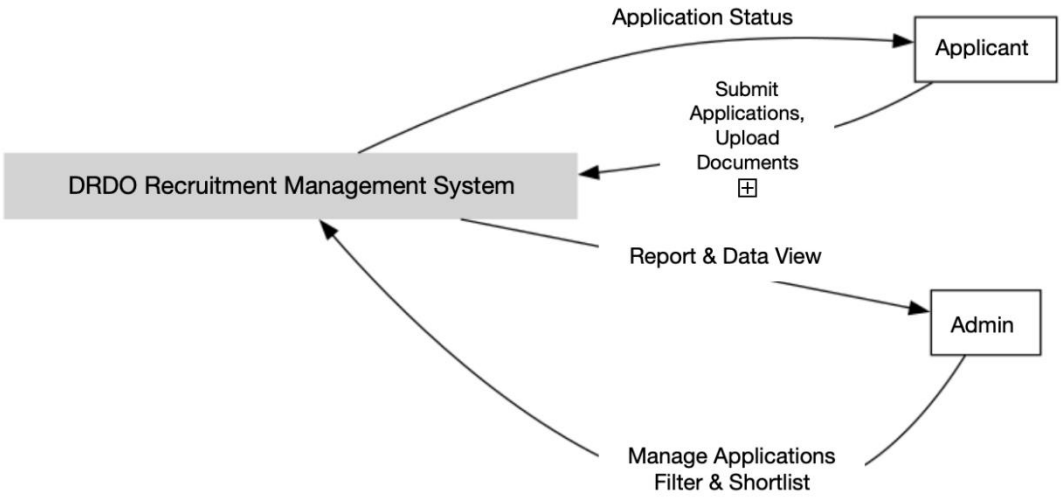
CHAPTER 5

SYSTEM DESIGN(SDS)

5.A. HIGH-LEVEL DESIGN

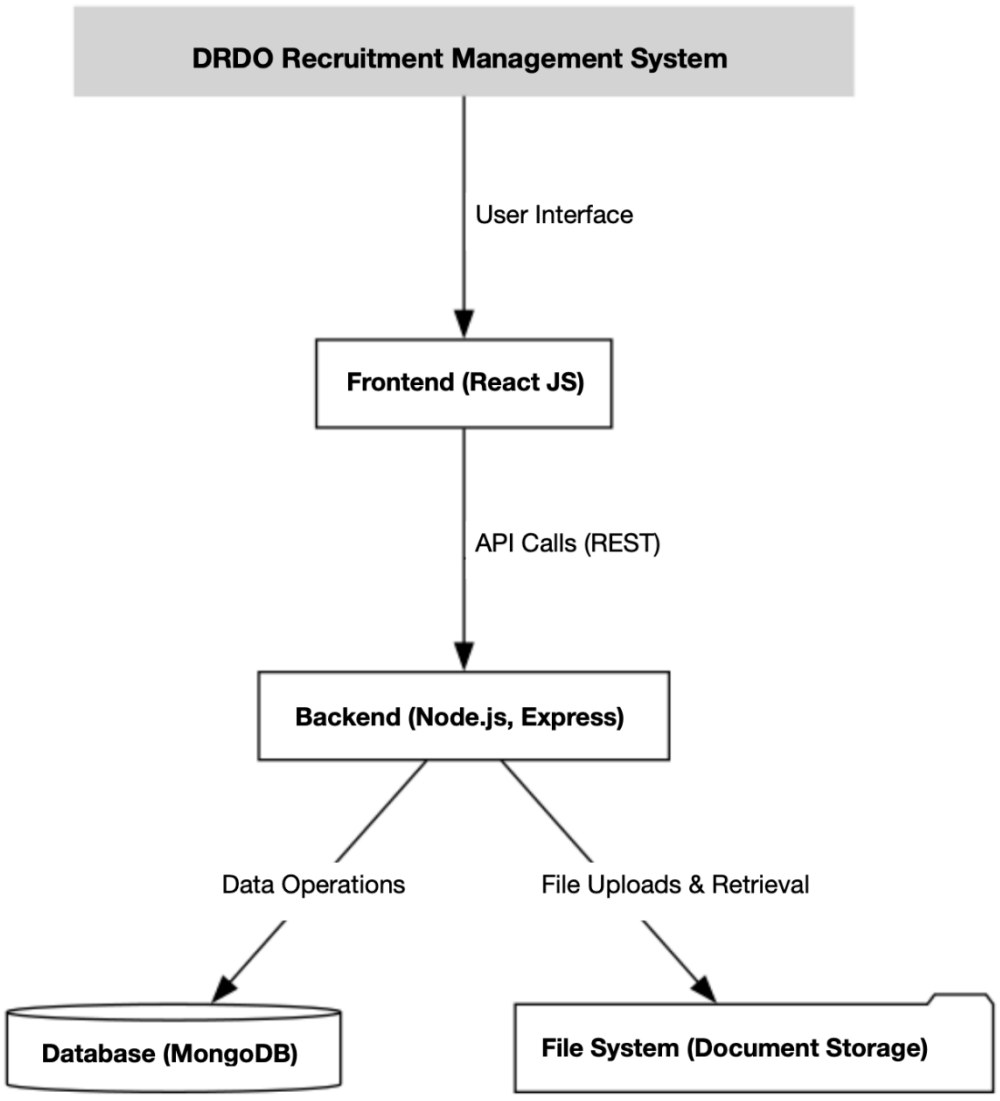
- **(Context Diagram)**

This context diagram for the DRDO Recruitment Management System provides a high-level overview of interactions between the system and its key actors: Applicant and Admin. Applicants can submit applications, upload documents, and check their application status, while Admins manage the system through report generation, data viewing, and filtering and shortlisting applications. It effectively highlights the flow of information between the users and the system.



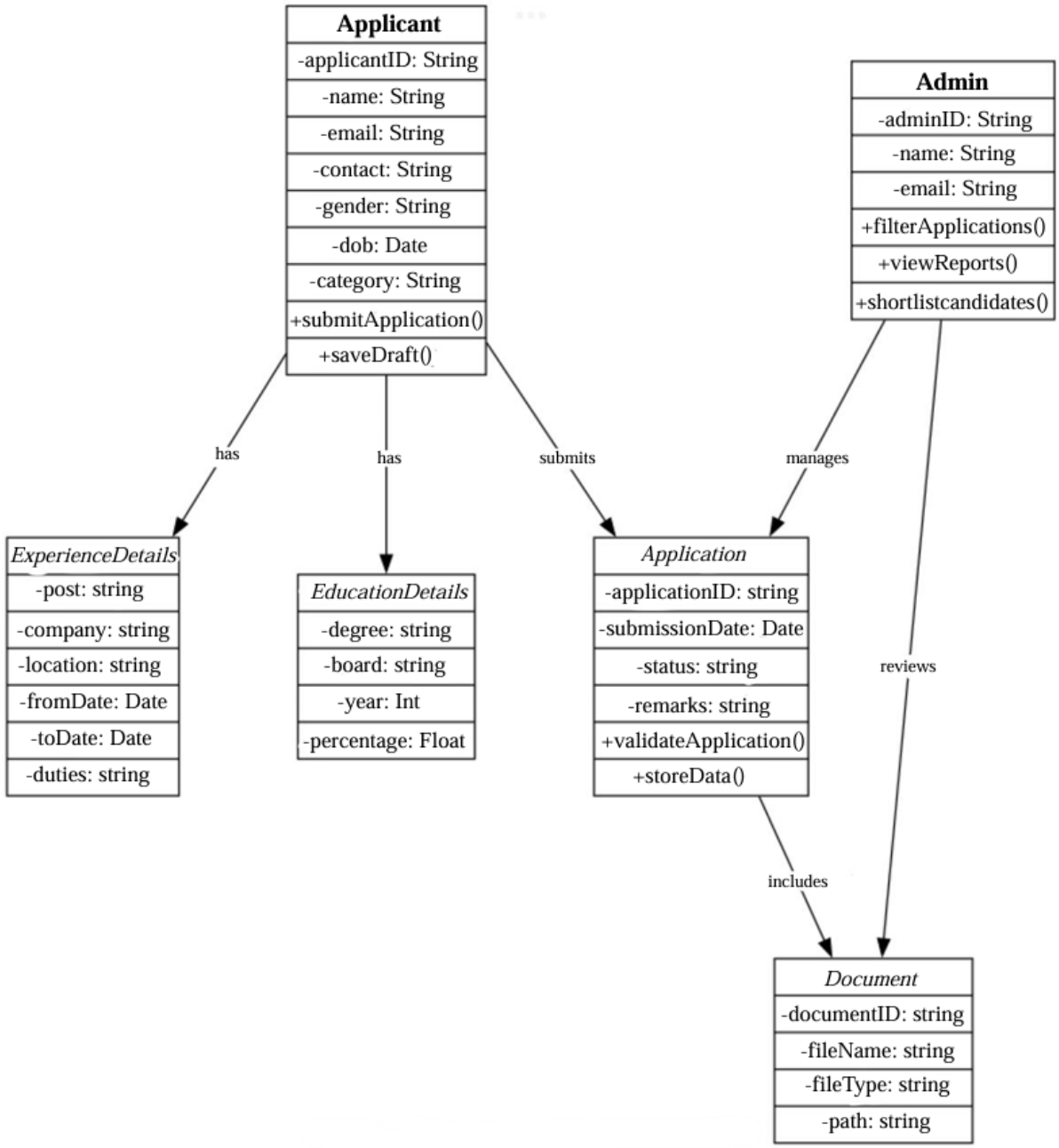
- **(Component Diagram)**

This component diagram represents the architecture of the DRDO Recruitment Management System. The Frontend built with React JS serves as the user interface, which interacts with the Backend (Node.js, Express) via RESTful API calls. The backend handles data operations through a MongoDB database and manages file uploads and retrieval using a file system for document storage. This architecture ensures a seamless flow of data and efficient system performance.



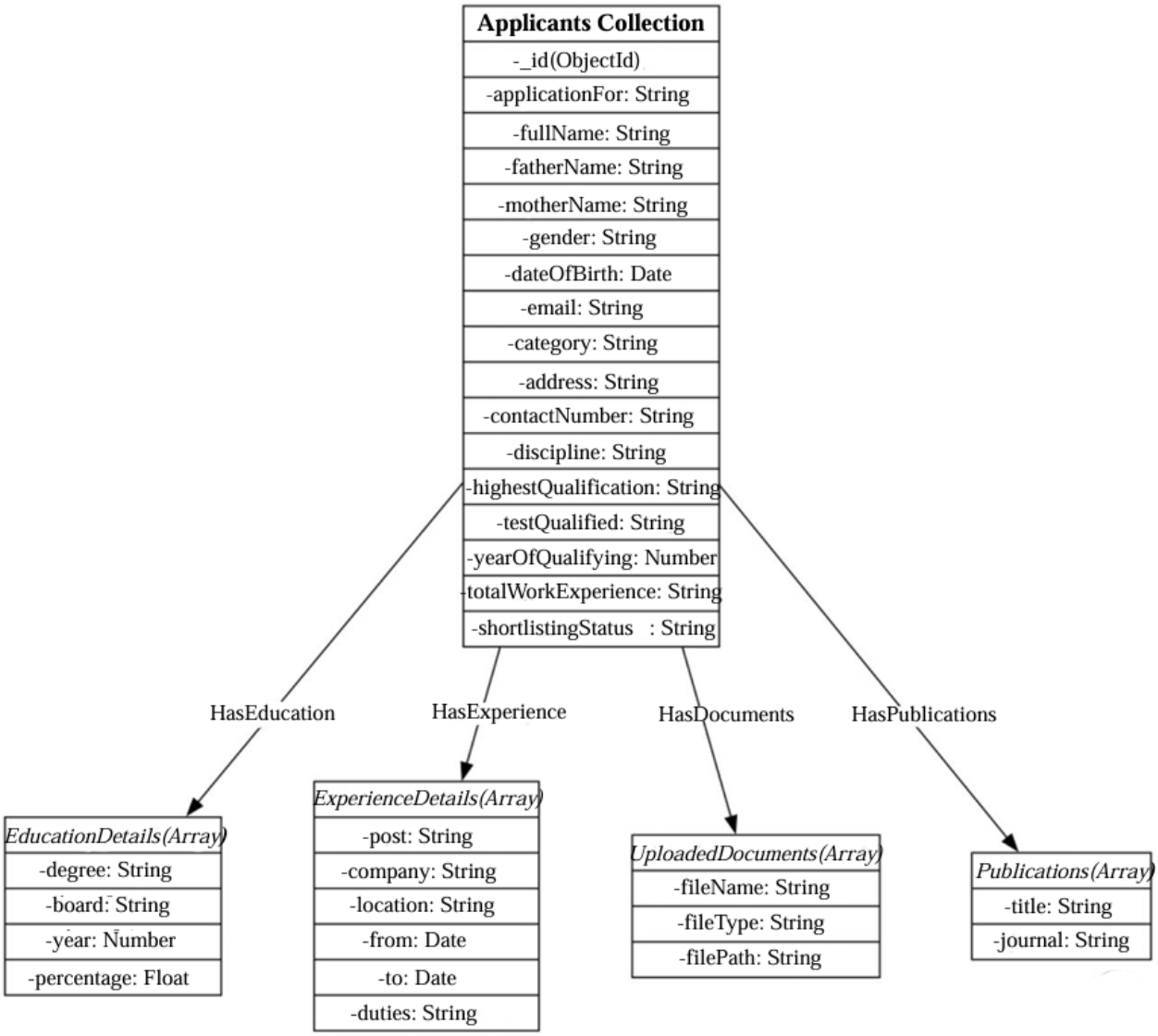
5.B. CLASS DIAGRAM

This class diagram represents the interaction between key entities in an application management system. The Applicant class submits applications containing education and experience details. Each Application includes documents and is reviewed by an Admin, who can filter, validate, and shortlist applications. The relationships highlight how Applicant details, experiences, and documents are managed, validated, and processed to streamline candidate shortlisting.



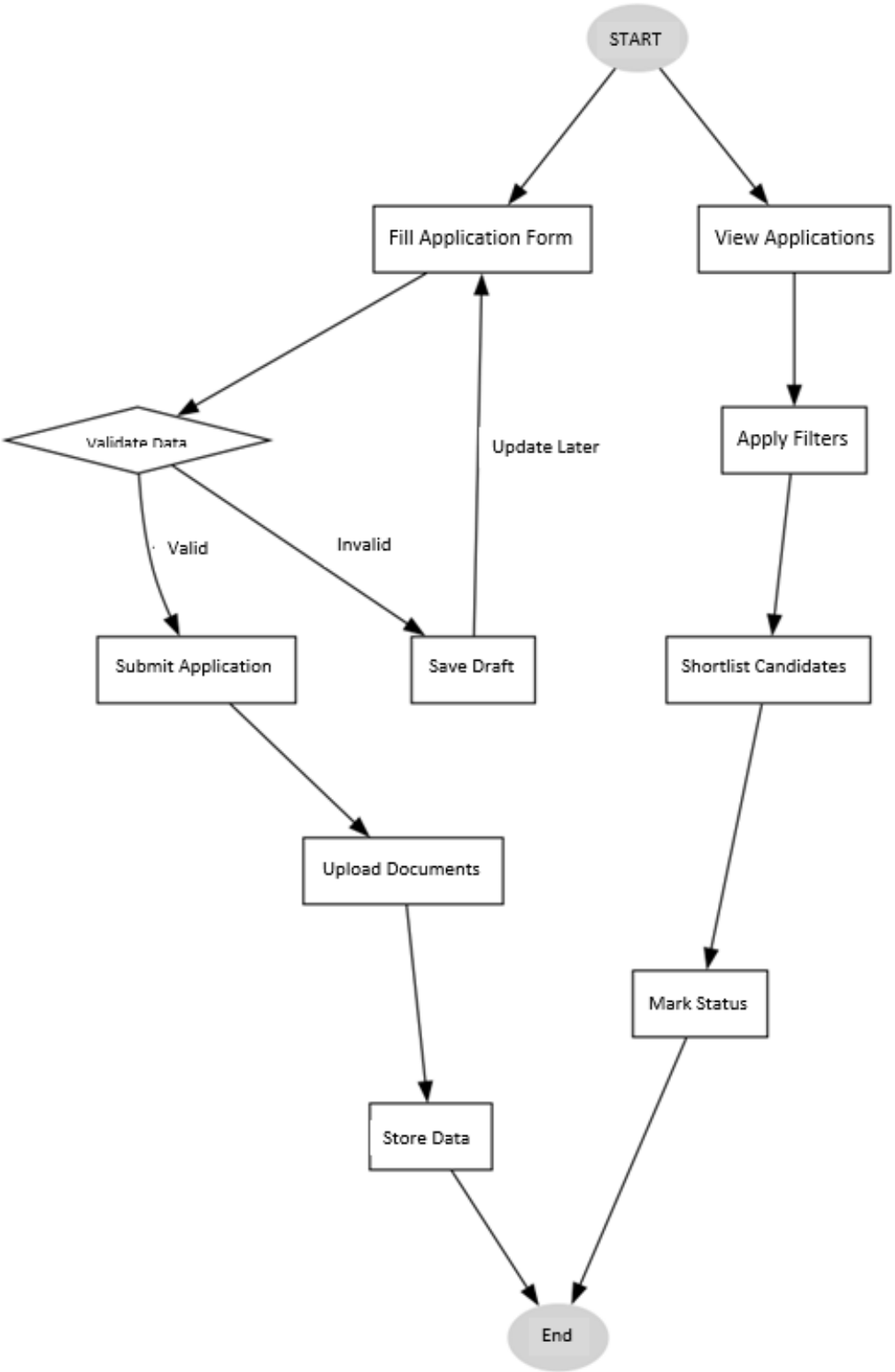
5.C. DATABASE DESIGN

This database design represents the structure of the Applicants Collection in a candidate management system. The main collection stores applicant details, such as personal information, qualifications, work experience, and application status. It has relationships with four arrays: EducationDetails, ExperienceDetails, UploadedDocuments, and Publications, enabling structured storage of academic records, professional experiences, uploaded files, and publications. This design ensures efficient organization and retrieval of applicant data for processing and shortlisting.



5.D. ACTIVITY DIAGRAM

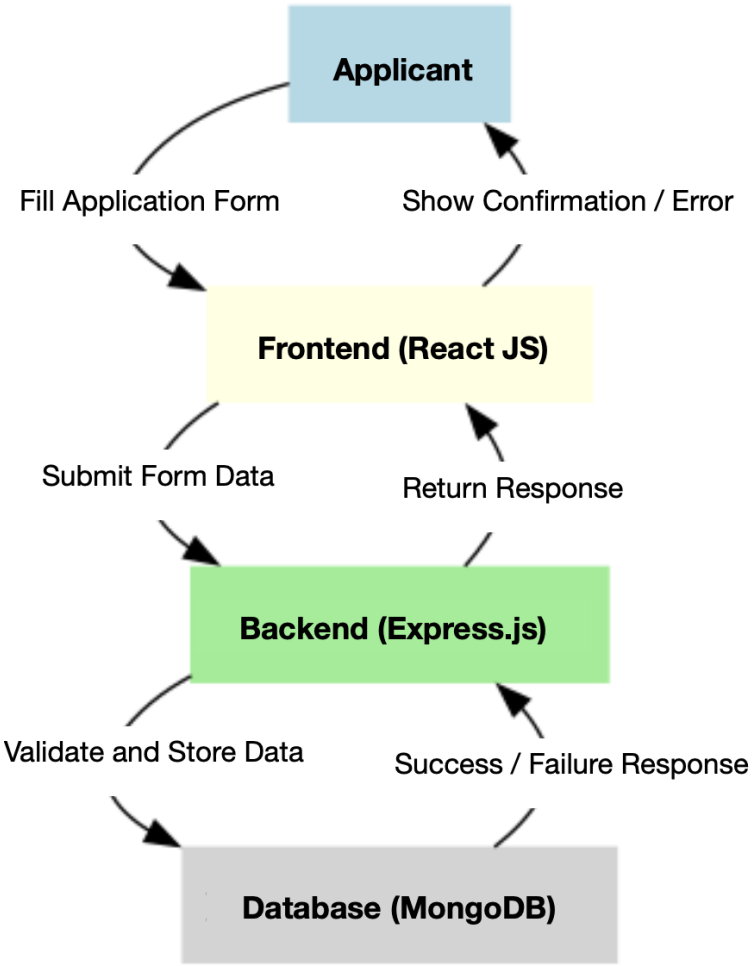
This **activity diagram** represents the workflow of the **DRDO Recruitment Management System**. It includes tasks like **filling the application form**, **validating data**, **submitting applications**, and **uploading documents**. Admins can **apply filters**, **shortlist candidates**, and **mark statuses**, leading to the final data storage and completion of the process.



5.E. SEQUENCE DIAGRAMS

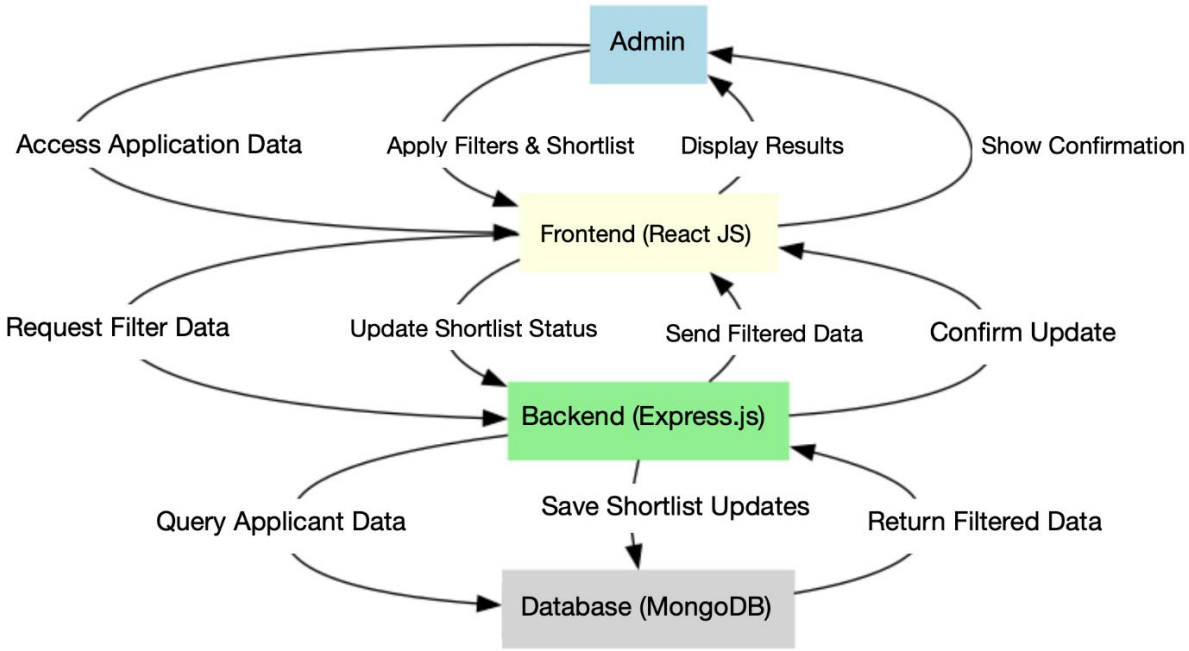
- **(Applicant)**

This context diagram outlines the flow of data between the Applicant, Frontend (React JS), Backend (Express.js), and Database (MongoDB) in the DRDO Recruitment Management System. The applicant fills the application form, which is submitted through the frontend to the backend for validation and storage. The backend interacts with the database and provides a success or failure response back to the frontend, ensuring a clear feedback loop for the user.



- **(Admin)**

This sequence diagram illustrates the flow of interactions in the DRDO Recruitment Management System between the Admin, Frontend (React JS), Backend (Express.js), and Database (MongoDB). The admin requests actions like filtering applications, updating shortlist status, and displaying results, which flow through the frontend and backend, interacting with the database for querying and saving data. The sequence ensures a streamlined data exchange and confirmation process.



CHAPTER 6

CODING

Frontend Modules

1. App.js

Handles state management and navigation for the multi-step form.

Key Features:

- Manages the state for all form steps.
- Provides navigation between steps and handles the final submission.

Code Snippet:

```jsx
import React, { useState } from 'react';
import PersonalInfo from './PersonalInfo';
import Qualifications from './Qualifications';
import DocumentsUpload from './DocumentsUpload';
import ReviewApply from './ReviewApply';

function App() {
  const [currentStep, setCurrentStep] = useState(1);
  const [formData, setFormData] = useState({
    personalInfo: {},
    qualifications: [],
    documents: [],
    declaration: false,
  });

  const handleNext = () => setCurrentStep((prev) => prev + 1);
  const handleBack = () => setCurrentStep((prev) => prev - 1);

  const handleSubmit = async () => {
    const response = await fetch('http://localhost:5001/api/submit', {
      method: 'POST',
      headers: { 'Content-Type': 'application/json' },
      body: JSON.stringify(formData),
    });
    console.log(await response.json());
  };

  return (
    <>
      {currentStep === 1 && <PersonalInfo data={formData} setData={setFormData} />}
      {currentStep === 2 && <Qualifications data={formData} setData={setFormData} />}
      {currentStep === 3 && <DocumentsUpload data={formData} setData={setFormData} />}
      {currentStep === 4 && <ReviewApply data={formData} onSubmit={handleSubmit} />}
      <button onClick={handleBack} disabled={currentStep === 1}>Back</button>
      <button onClick={handleNext} disabled={currentStep === 4}>Next</button>
    </>
  );
}

export default App;
```

2. PersonalInfo.js

Handles input fields for personal details and basic validations.

Key Features:

- Validates required fields (e.g., name, email, contact number).
- Allows photo upload.

Code Snippet:

```jsx
import React from 'react';

function PersonalInfo({ data, setData }) {
  const handleChange = (e) => {
    const { name, value } = e.target;
    setData((prev) => ({ ...prev, personalInfo: { ...prev.personalInfo, [name]: value } }));
  };

  return (
    <form>
      <label>Name: <input name="fullName" onChange={handleChange} /></label>
      <label>Email: <input name="email" onChange={handleChange} /></label>
      <label>Photo: <input type="file" name="photo" accept="image/*" /></label>
    </form>
  );
}

export default PersonalInfo;
```

Backend Modules

1. server.js

The backend handles API routes, form validation, and file uploads**.**

Key Features:

- Connects to MongoDB to store applicant data.
- Uses Multer for file uploads.
- Provides REST APIs for submitting and fetching data.

Code Snippet:

```
const express = require('express');
const mongoose = require('mongoose');
const multer = require('multer');
const app = express();

app.use(express.json());
mongoose.connect('mongodb://localhost:27017/drdo_recruitment');

// Define Schema
const applicantSchema = new mongoose.Schema({
  fullName: String,
  email: String,
  personalInfo: Object,
  qualifications: Array,
  documents: Array,
  declaration: Boolean,
});
const Applicant = mongoose.model('Applicant', applicantSchema);

// Configure Multer
const upload = multer({ dest: 'uploads/' });

// API Endpoint: Submit Form Data
app.post('/api/submit', upload.fields([{ name: 'photo' }, { name: 'documents' }]), async (req, res) => {
  const formData = JSON.parse(req.body.formData);
  const newApplicant = new Applicant(formData);
  await newApplicant.save();
  res.status(201).json({ message: 'Application submitted successfully' });
});

// API Endpoint: Get Applicants
app.get('/api/getApplicants', async (req, res) => {
  const applicants = await Applicant.find();
  res.json(applicants);
});

app.listen(5001, () => console.log('Server running on http://localhost:5001'));
```

Database Schema

Collection: FormData

Key Fields:

CHAPTER 7

TESTING

Testing ensures the application functions as expected, providing a seamless experience for applicants and admins. Based on your project, the following testing categories and test cases are proposed:

1. User Interface Testing

Objective: Validate the frontend's layout, consistency, and user experience.

Test Cases:

1. **Content Accuracy**:
    - Ensure all labels, headings, and messages are free from spelling or grammatical errors.
2. **Error Messages**:
    - Validate error messages for fields like "Full Name" or "Email" for correctness and relevance.
3. **Layout and Design**:
    - Check spacing between form fields, labels, and buttons for readability.
4. **Button Standardization**:
    - Verify that "Next," "Back," and "Submit" buttons are consistent in size, color, and behavior.
5. **Date Format Consistency**:
    - Ensure dates follow the same format (e.g., DD/MM/YYYY) across all screens.
6. **Responsiveness**:
    - Test the form layout on devices like desktops, tablets, and mobiles.

2. Functional Testing

Objective: Validate core functionalities of the system.

Test Cases:

1. **Field Validation**:
    - Ensure mandatory fields (e.g., "Full Name," "Email") highlight errors when left blank.
2. **File Uploads**:
    - Validate photo uploads accept only JPEG/PNG and display error messages for invalid file formats.
    - Ensure document uploads handle large files gracefully.
3. **Numeric Field Constraints**:
    - Verify numeric fields (e.g., "Contact Number") do not accept alphabets and enforce length constraints.
4. **Auto-populated Fields**:
    - Check pre-filled fields like applicant details in the "Review" section.
5. **Editable and Non-Editable Fields**:
    - Verify fields that should be read-only (e.g., auto-generated IDs) cannot be modified.
6. **Optional Fields**:
    - Confirm no errors are displayed for optional fields left empty.
7. **Confirmation Messages**:
    - Ensure success messages display after form submission or admin actions like shortlisting.

3. Role-Based Access Testing

Objective: Ensure role-specific features function correctly.

Test Cases:

1. **Admin Access**:
    - Verify that the admin can:
        - View all applicants.
        - Apply filters like qualifications or experience.
        - Mark applicants as shortlisted or rejected.

2. **Applicant Access**:
    - Confirm applicants can only:
        - Fill and submit the form.
        - View their own submission status.
3. **Unauthorized Access**:
    - Ensure restricted features, like the admin dashboard, cannot be accessed by applicants.

4. Database Testing

Objective: Validate data integrity in MongoDB.

Test Cases:

1. **Data Storage**:
    - Verify form data is stored correctly in the formdatas collection.
2. **File Paths**:
    - Check that uploaded file paths are saved correctly.
3. **Filter Results**:
    - Ensure the database query returns accurate results when filters are applied.

5. Performance Testing

Objective: Ensure the application performs well under various conditions.

Test Cases:

1. **Load Testing**:
    - Submit multiple forms simultaneously to test server response time.
2. **Database Query Performance**:
    - Verify database performance when fetching a large number of applicants.
3. **Form Navigation**:
    - Ensure no delays occur when navigating between form steps.

Summary Table of Test Cases:

Test Case ID	Category	Test Scenario	Expected Outcome
TC-UI-01	User Interface	Validate "Full Name" field label for correctness	Label reads "Full Name (Block Letters Only)"
TC-FN-01	Functional	Submit form without mandatory fields	Error: "Full Name is required"
TC-DB-01	Database	Verify photo file path in database	Path stored as /uploads/photo.jpg
TC-PF-01	Performance	Submit 100 forms simultaneously	Average response time < 2 seconds

CHAPTER 8

USER INTERFACE

DRDO Vacancy Form

The application process begins with the Personal Information page, where applicants provide their basic details. This is the first step of a multi-step form, displayed in a clear and organized manner. The form is structured as a multi-step process with a sidebar menu displaying steps like Personal Information, Qualifications, Self-Declaration and Review & Apply.

Users cannot proceed to the next steps without completing the current page and addressing any validation errors. This ensures proper data entry and seamless navigation through the application process.

The overall interface has a clean layout and is easy to understand, ensuring applicants can fill out the required details efficiently.

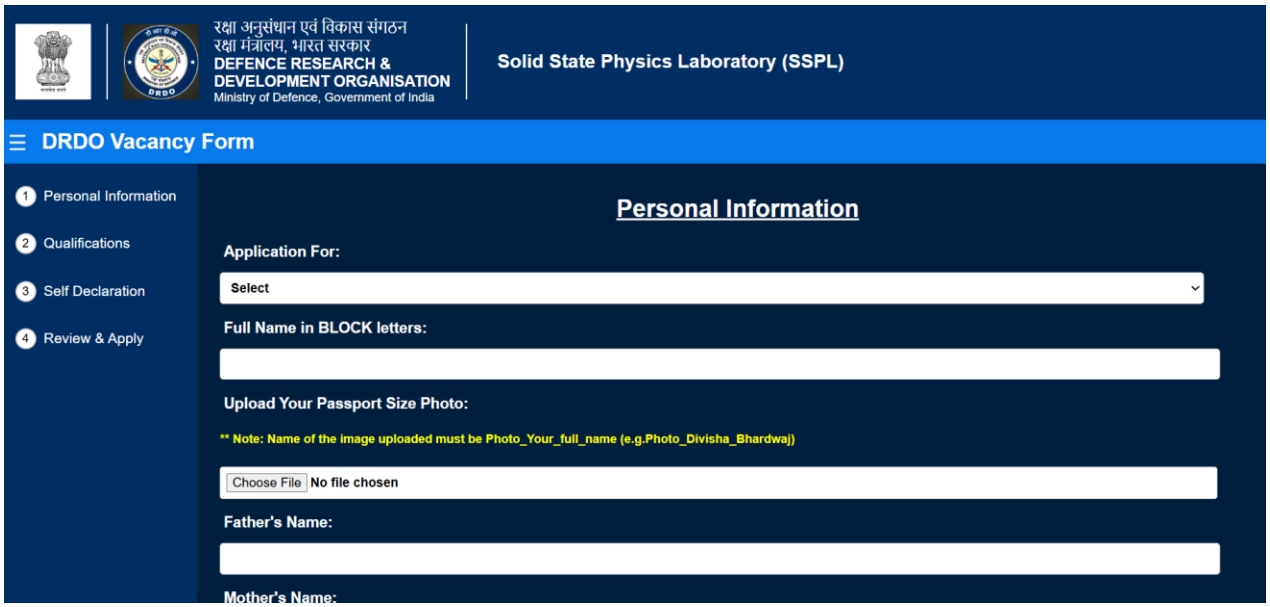
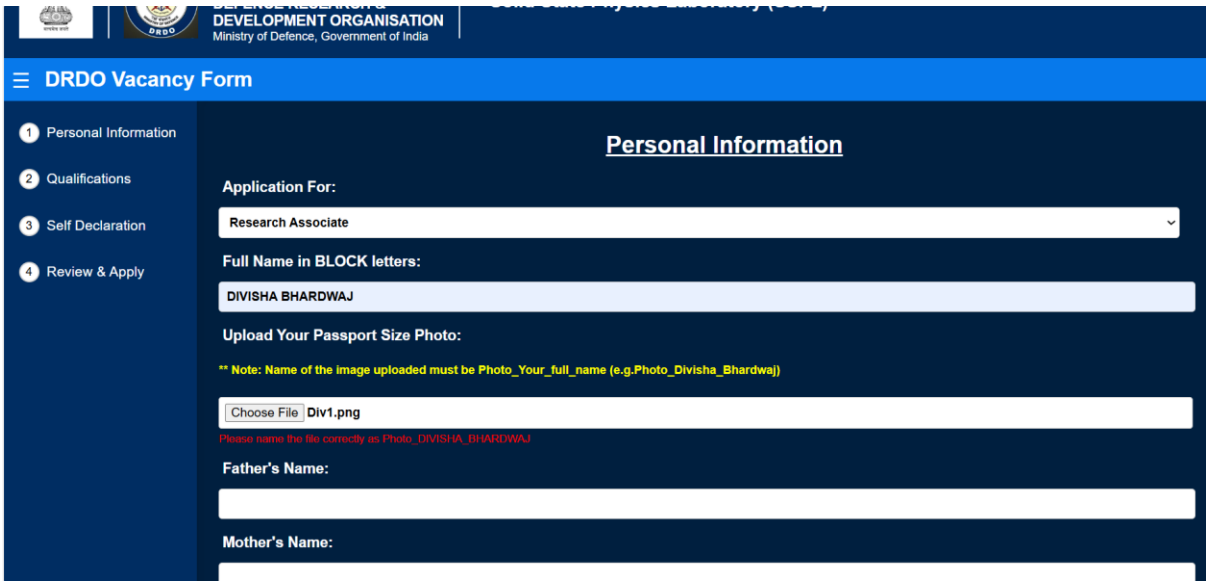
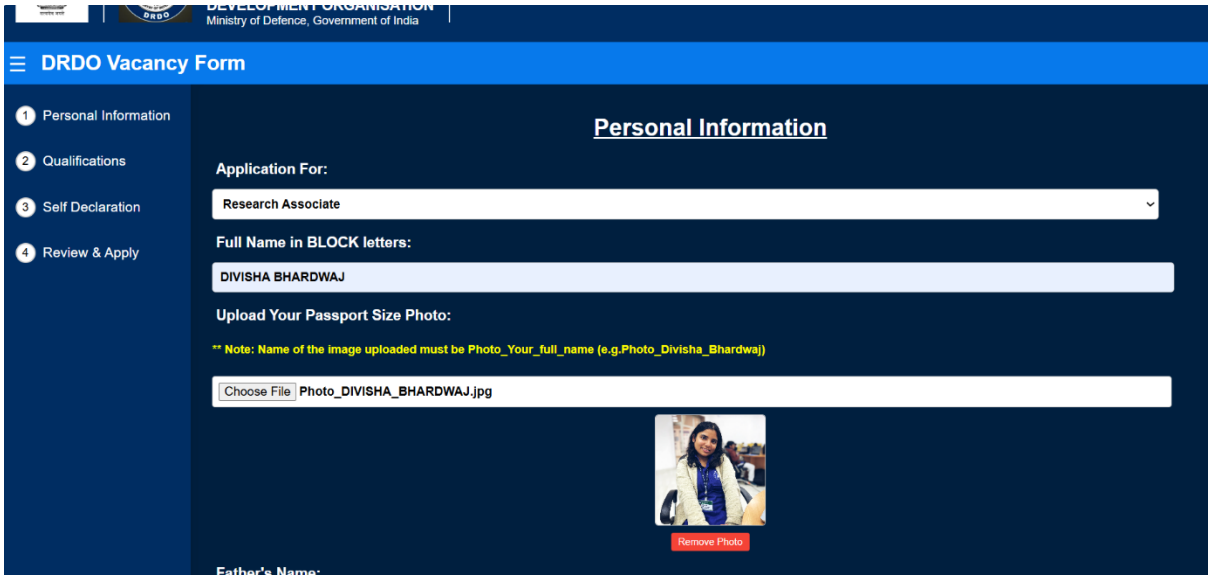


Photo Upload with Validation Check

This page highlights the photo upload functionality within the form. The system enforces a specific file naming convention to ensure uniformity and proper identification. If the uploaded photo does not adhere to the required format (e.g., Photo_FullName), an error message is displayed, prompting the user to correct the filename.



Upon successful upload, the system provides a Remove Photo option for user convenience to change or edit photo.



Field Validation for Accuracy

The system implements robust field validation to ensure accurate and complete data entry. Required fields like Email and Contact Numbers are validated in real-time. For example:

- Email: Displays an error if left blank or entered incorrectly.
- Contact Number: Requires exactly 10 digits; any deviation triggers an error message.

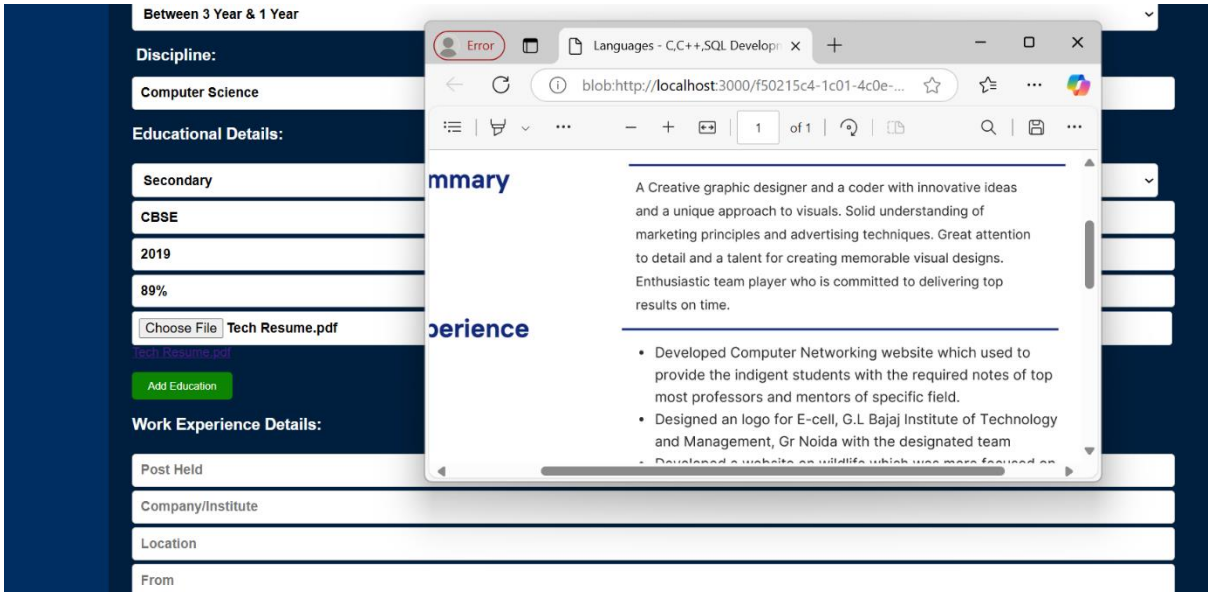
Users cannot proceed to the next step by clicking "Next" until all data on the current and previous pages are correctly filled. This validation ensures error-free form submission and enhances data integrity.

Navigation and Progress Tracking

Upon successfully filling and validating all fields, users can proceed to the next page. A green tick appears on the menu bar for completed sections, providing clear visual feedback on progress.

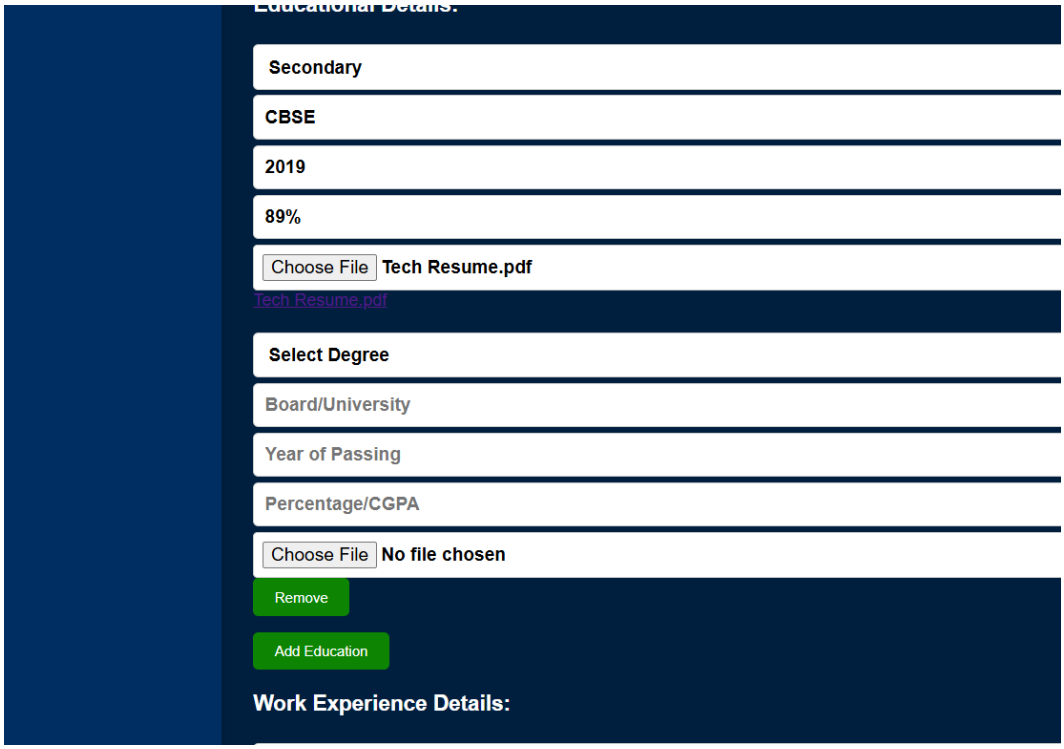
Uploaded File Preview

The system allows users to preview uploaded files, such as mark sheets or resumes, by clicking on the file name. This opens the document in a new window, ensuring users can verify the correct file before proceeding.



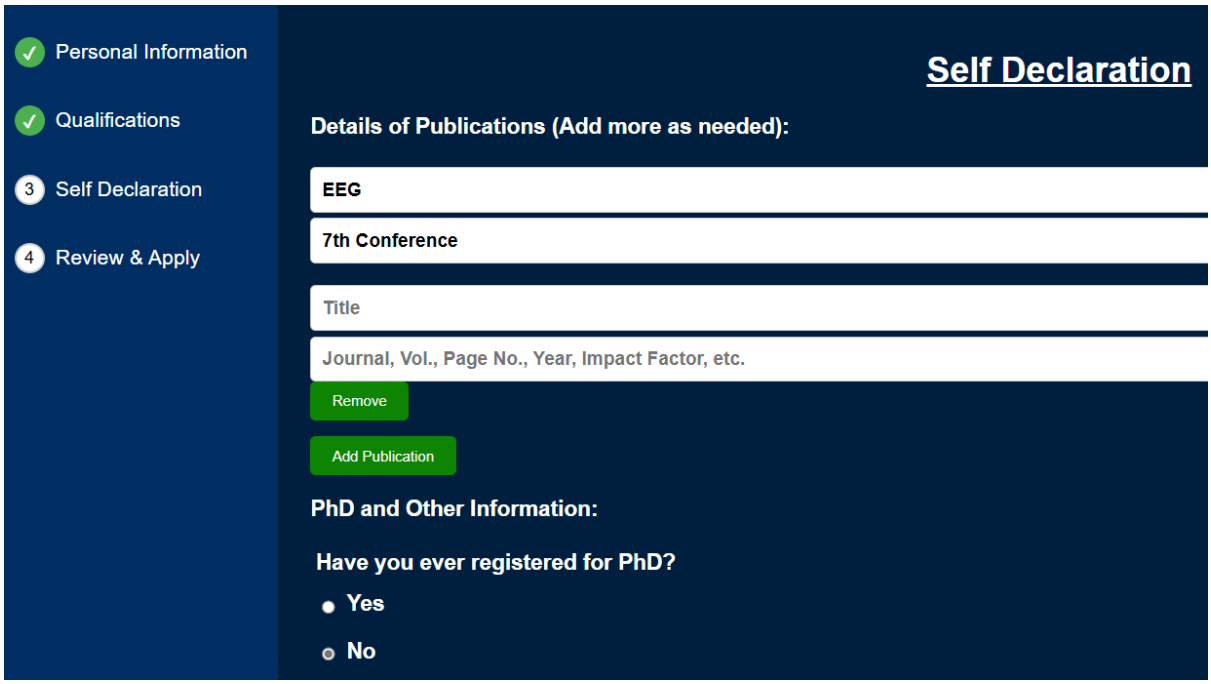
Add and Remove Education/Work Experience

The system provides functionalities to add or remove **Educational Details** and **Work Experience** dynamically. Users can add multiple entries for degrees, boards, percentages, and work details as needed. The "Remove" button ensures flexibility by allowing users to delete unwanted entries seamlessly.

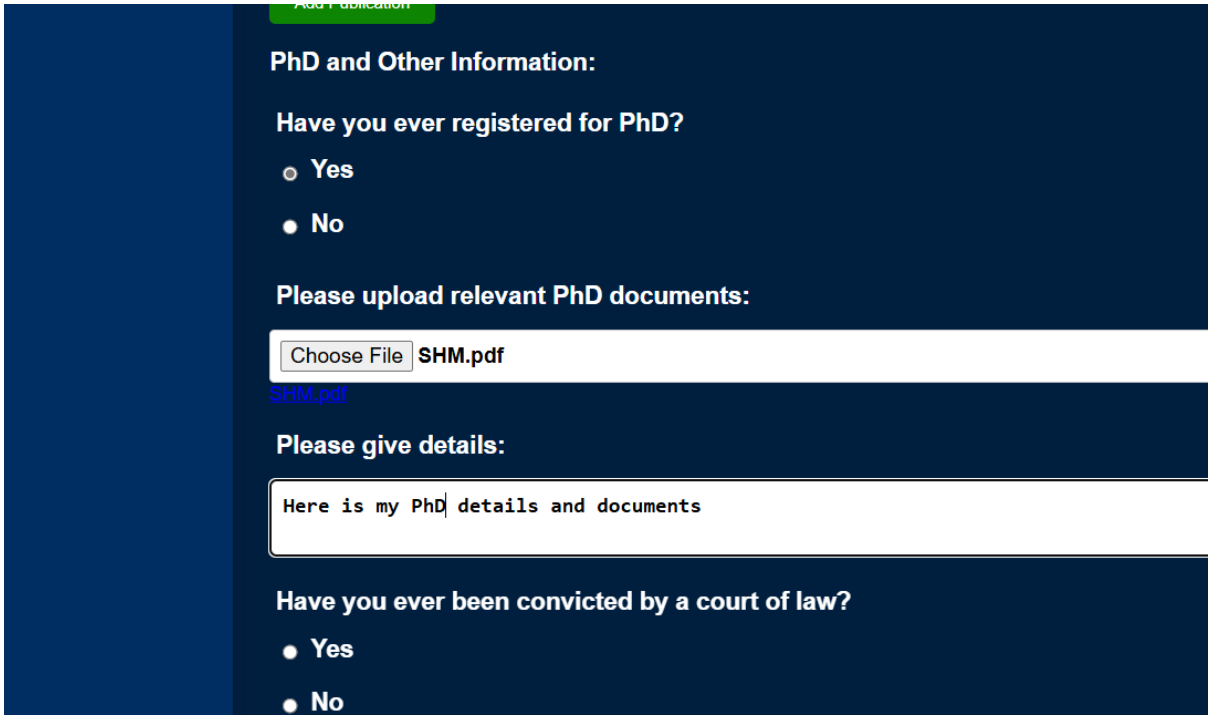


PhD and Other Information Page

If **"No"** is selected for PhD registration, users can proceed.

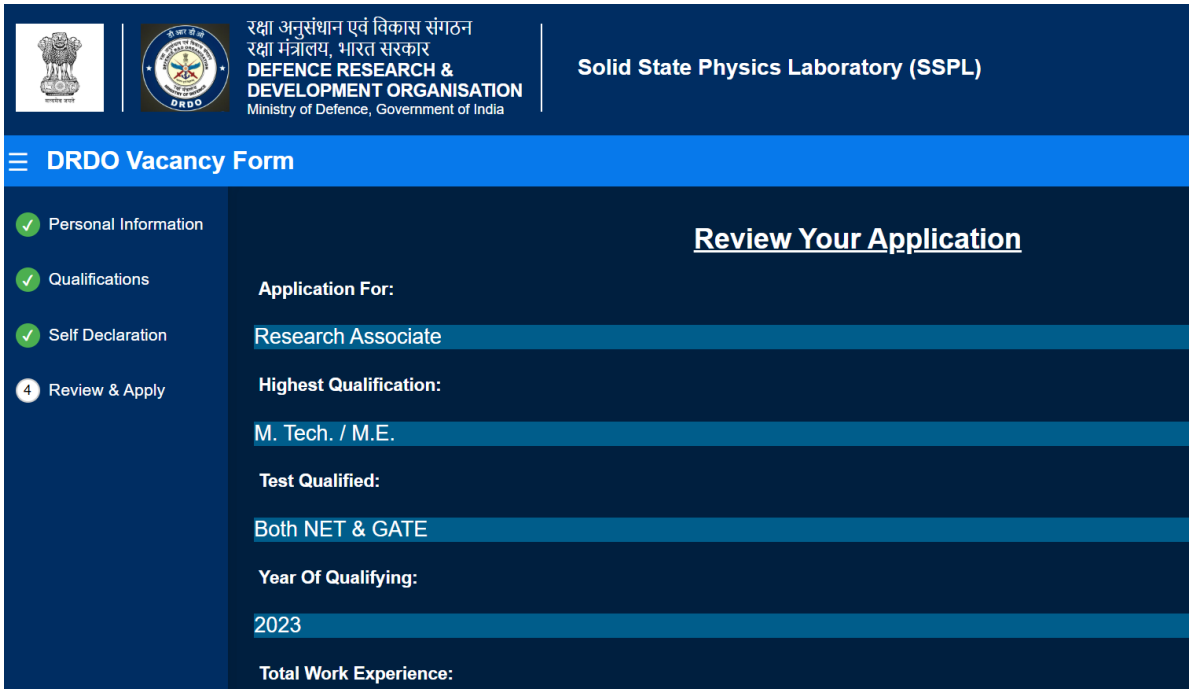


If **"Yes"** is selected, additional fields appear to upload **PhD documents** and provide details.For legal information, fields appear based on **Yes/No** selections, ensuring only relevant inputs are required.



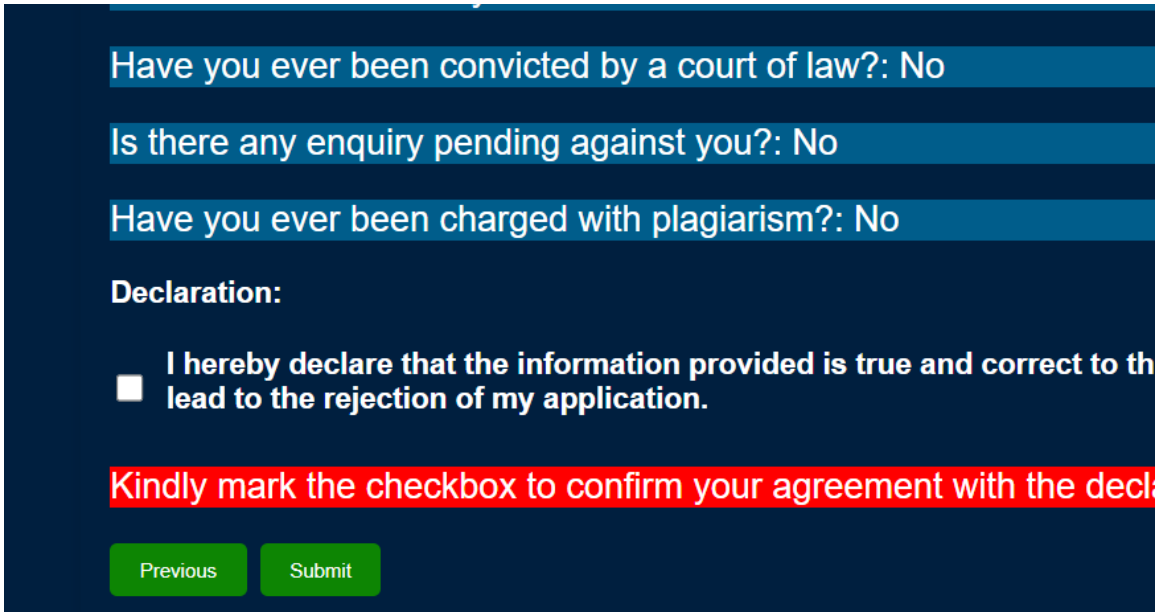
Review Your Application Page

This section displays all the data entered by the applicant for verification before final submission. Users can review their information, and if any changes are needed, they can go back to the respective sections and update the details. This ensures accuracy before submission.



Declaration Agreement

Before submitting the application, the user must agree to the declaration by checking the provided checkbox. If the checkbox is not marked, an error or warning appears, prompting the user to confirm their agreement. This ensures the accuracy and authenticity of the submitted information.

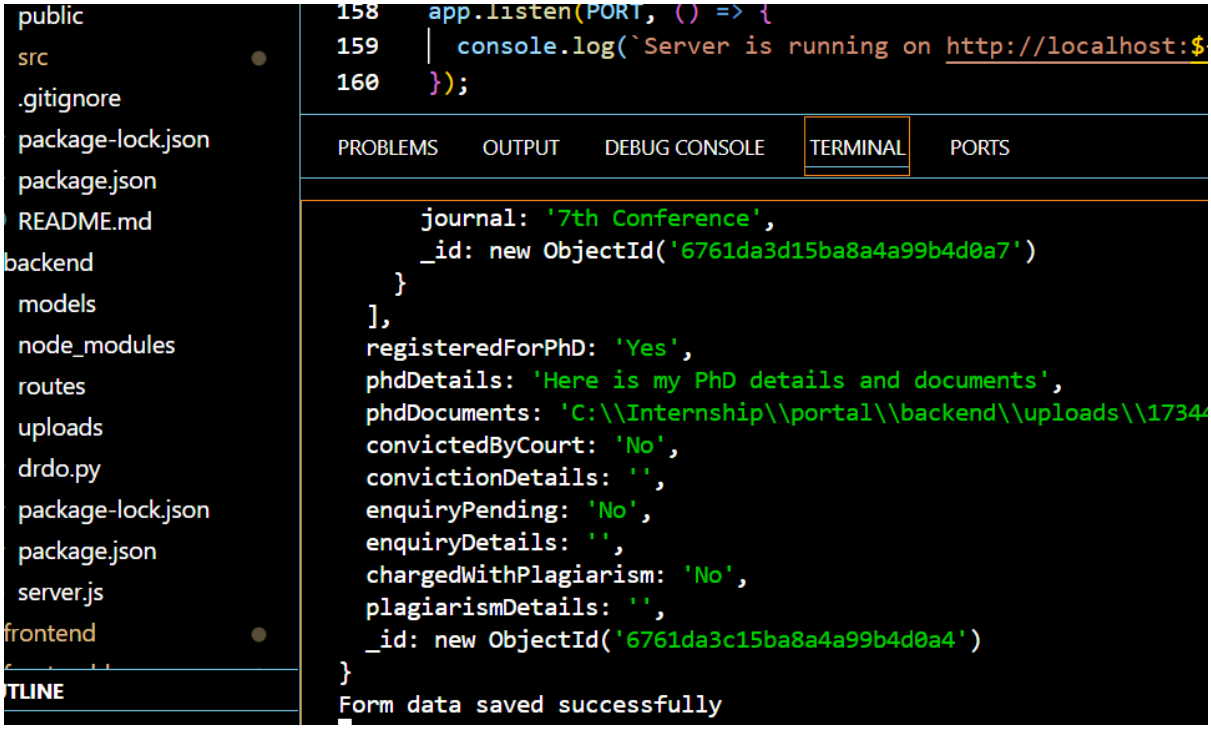


Application Submission Confirmation

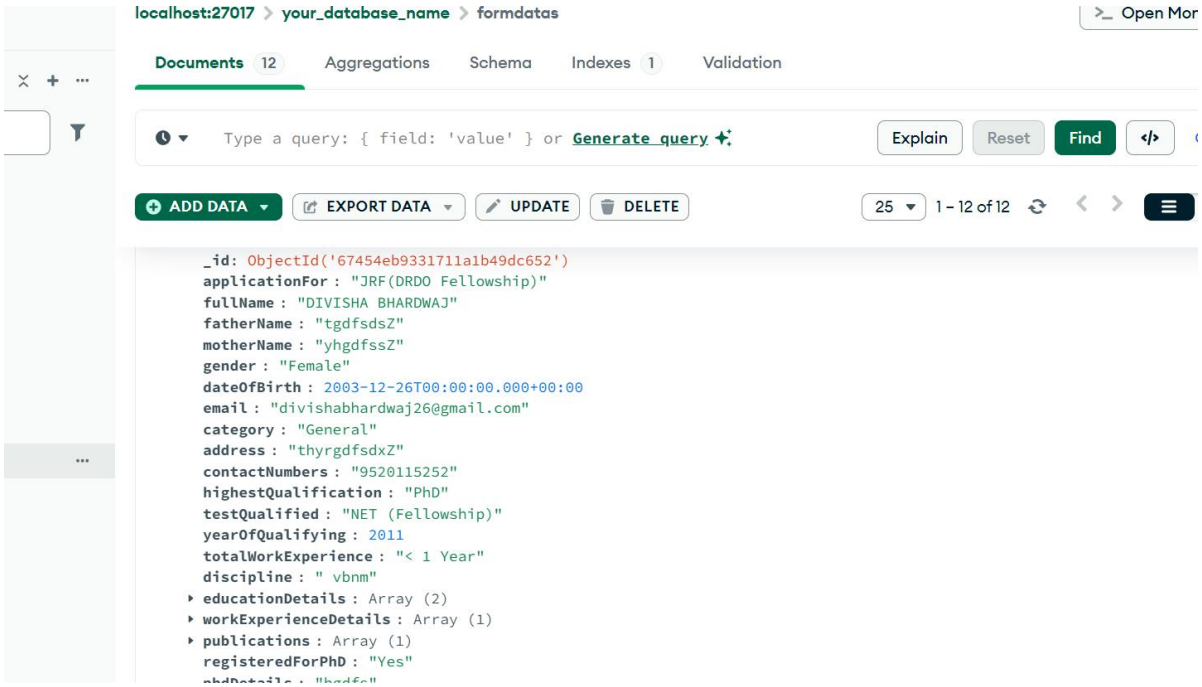
After successfully submitting the application, a confirmation page appears stating "Application Submitted Successfully." This page provides assurance to the user that their application has been recorded in the system and no further action is needed at this stage.

Backend Storage

On the backend, the data is saved into the MongoDB database, and a success log, "Form data saved successfully", appears on the server terminal.

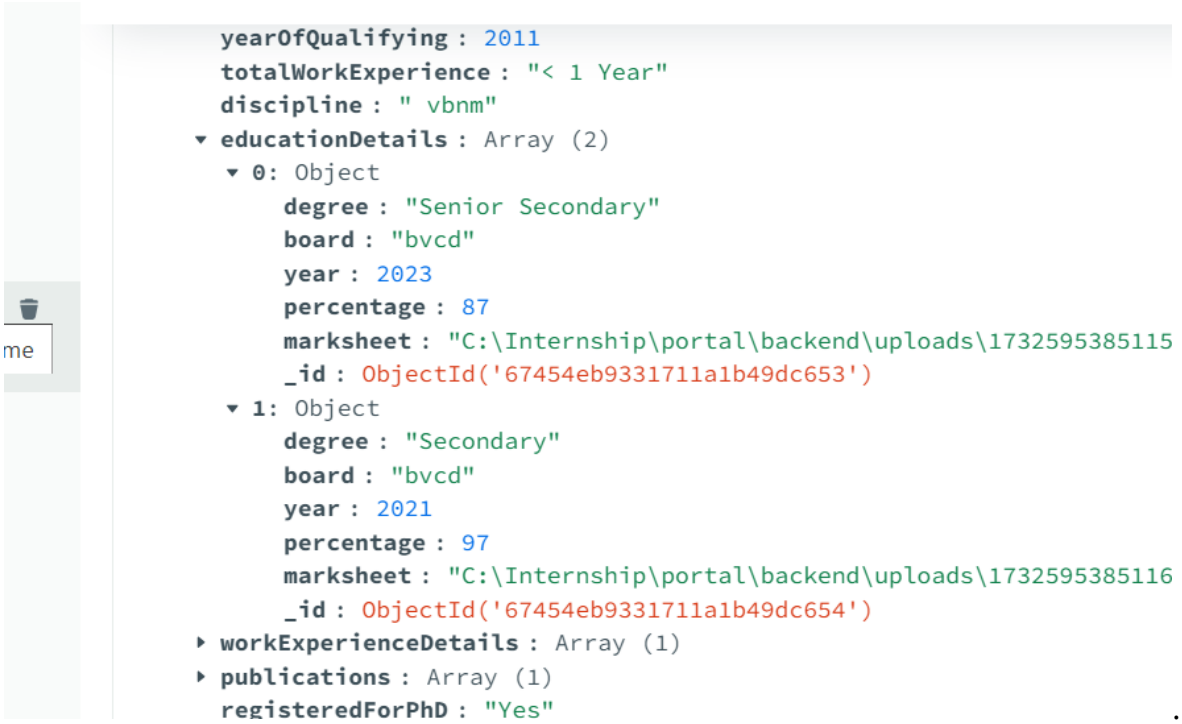


The database stores all entered details, including personal information, qualifications, work experience, and other relevant fields



Handling Multiple Entries

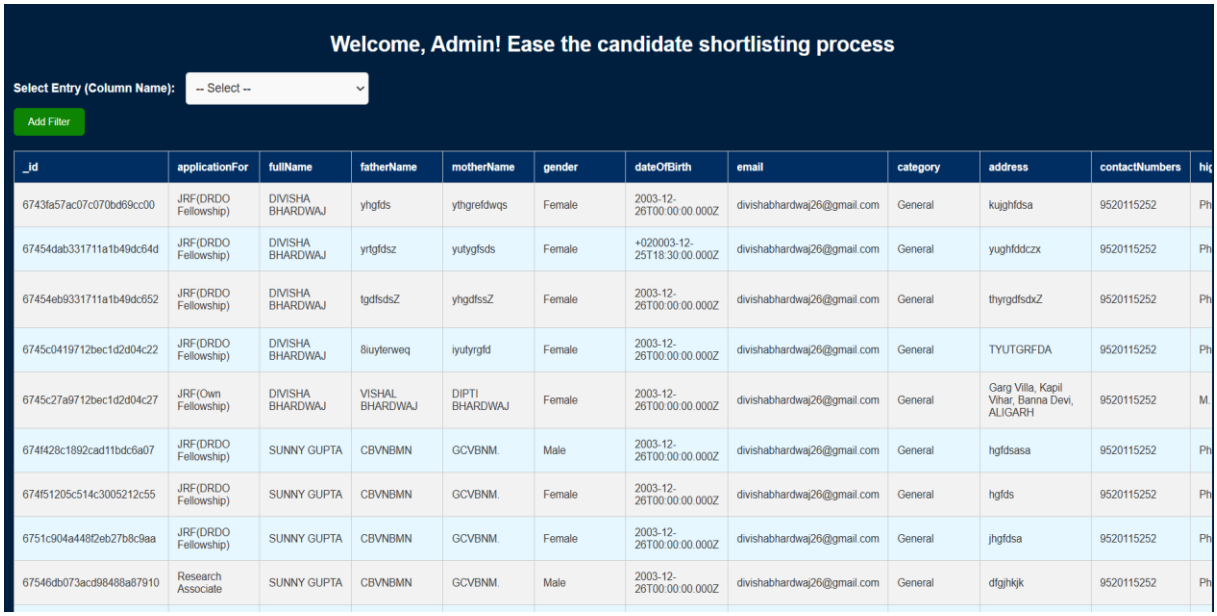
For fields that allow multiple entries, such as **Education Details** and **Work Experience**, the backend stores them as **arrays of objects** in the database. Each object represents a separate entry containing attributes like degree, board, year, percentage, and uploaded files. This approach ensures organized storage and easy retrieval for dynamic data.



Admin Dashboard

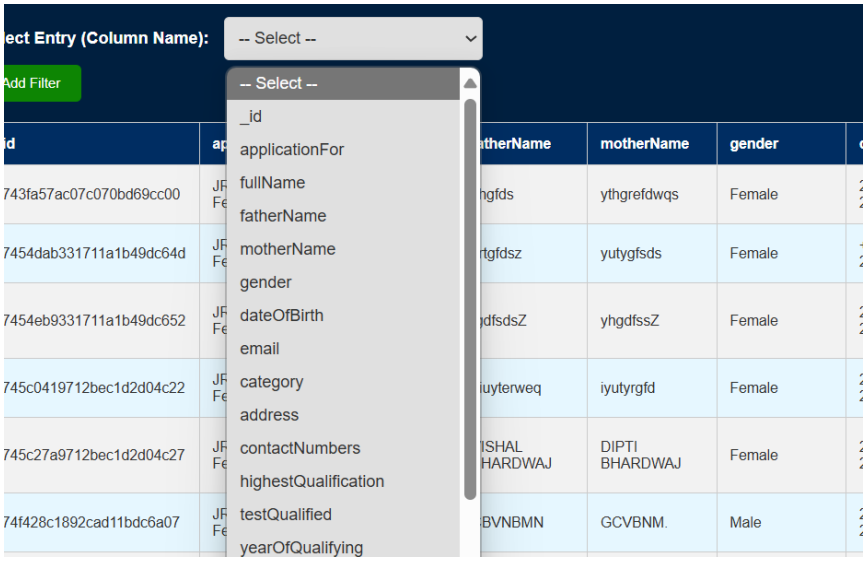
The **Admin Dashboard** displays all the submitted applications in a tabular format. Each row contains detailed candidate information, such as name, gender, date of birth, qualifications, and contact details.

- Admin can **filter data** using column names for efficient shortlisting.
- The dashboard is user-friendly, allowing the admin to **view, analyze, and shortlist candidates** based on specific criteria.



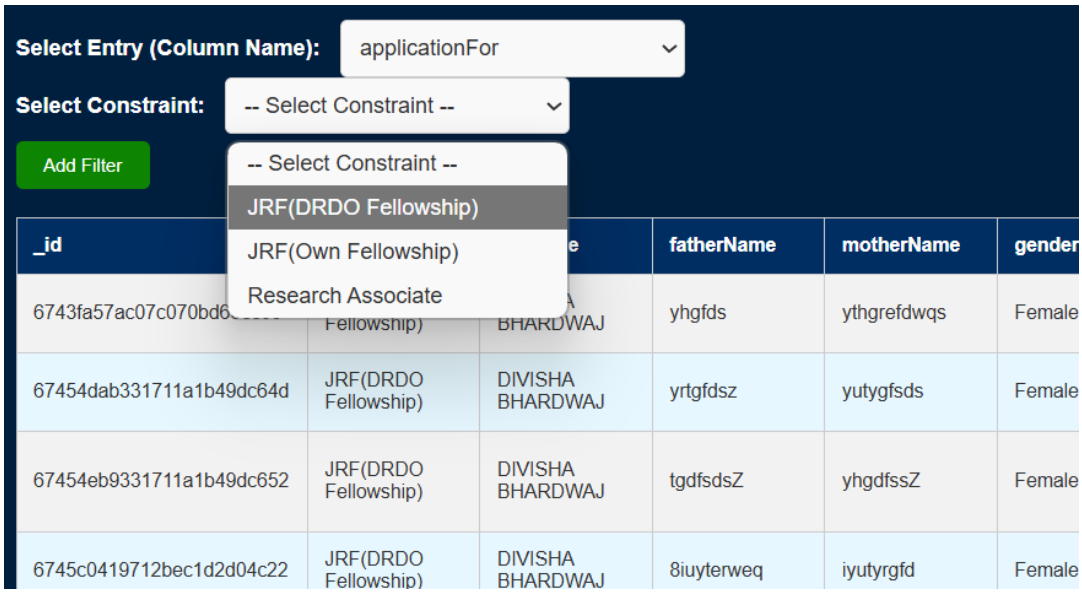
Admin Filter Dropdown

The dropdown allows the admin to select specific fields from the backend to apply filters. This helps in shortlisting candidates quickly based on labels like email, category, or yearOfQualifying**.**



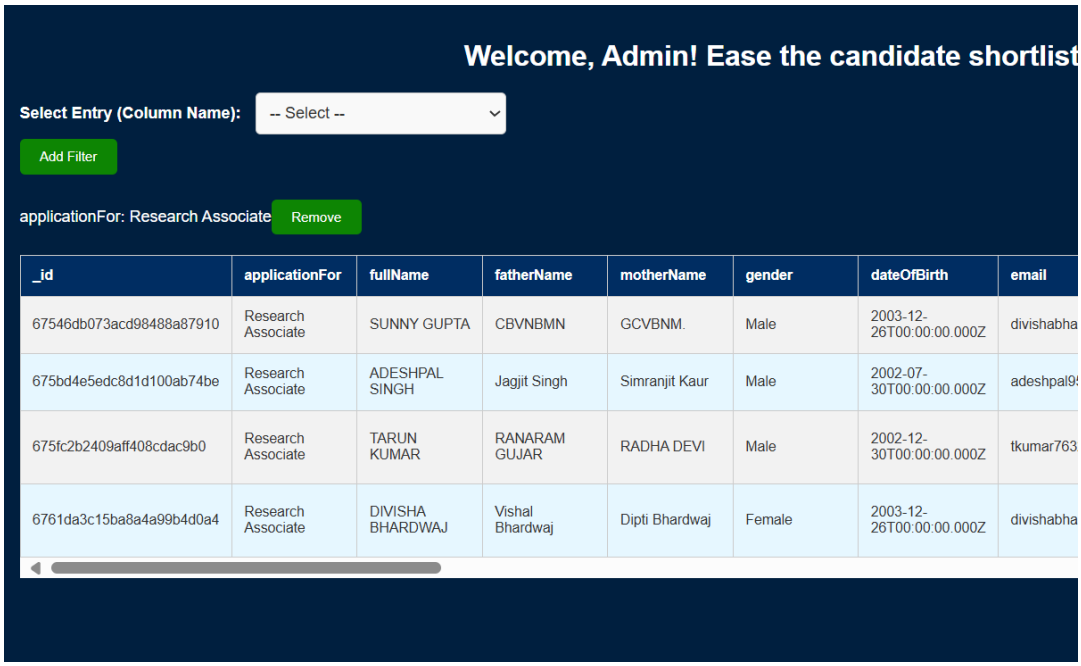
Constraint Selection

Once a column label is selected, a second dropdown appears to set specific constraints for that field. For example, selecting applicationFor allows filtering candidates for "JRF(DRDO Fellowship)", "JRF(Own Fellowship)", or "Research Associate". This ensures precise and efficient candidate shortlisting.



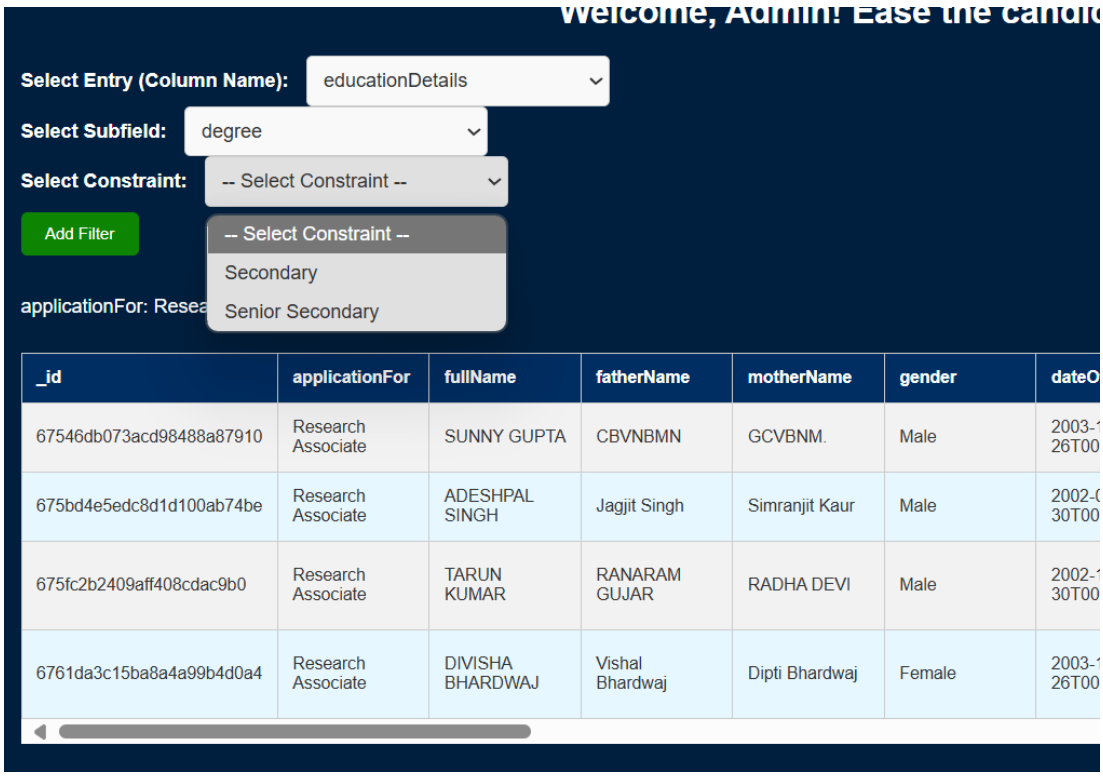
Filter Application and Results

After selecting the desired column and constraint, clicking "Add Filter" triggers a query at the backend. The system processes the filter against the database and displays the eligible candidates matching the criteria on the table. This makes shortlisting efficient and dynamic for the admin.



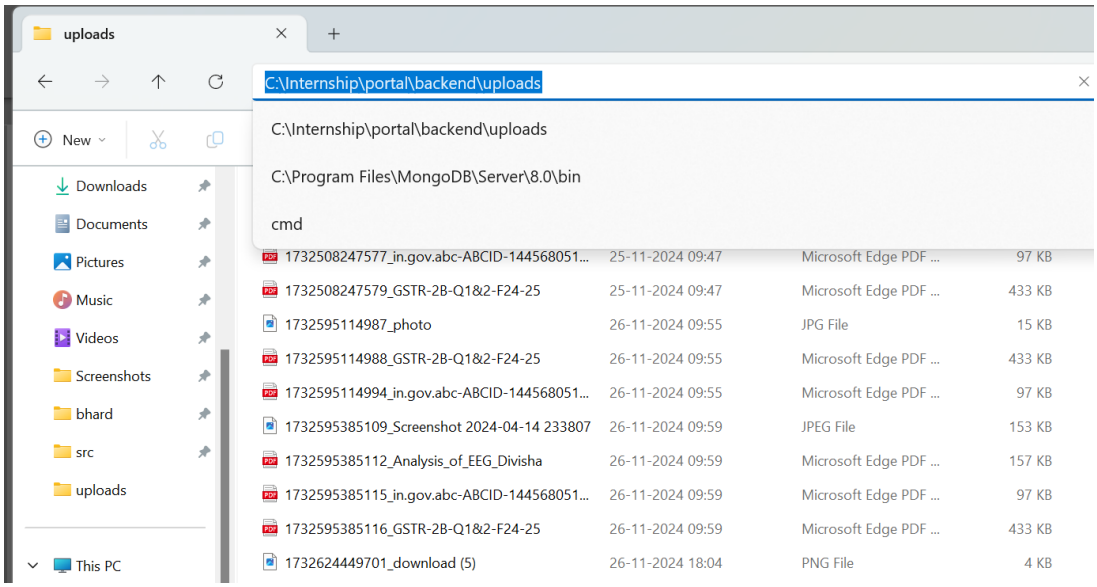
Filtering on Nested Fields

For complex fields like education details or work experience, the admin selects a subfield (e.g., degree, board) before applying constraints. This ensures accurate filtering based on specific criteria, refining candidate results dynamically.



Upload Folder Management

All uploaded documents, PDFs, and images are stored in a dedicated folder named uploads. Each file is saved with a unique ID as a prefix for easy identification and retrieval. This ensures clear file organization and quick access when needed.



CHAPTER 9

APPENDICES

Appendix A: Project Modules Overview

This appendix provides an overview of the main components developed during the project.

1. **Frontend (React.js):**
    - Designed an interactive and responsive user interface.
    - Managed forms for **personal details**, **educational qualifications**, **work experience**, and **self-declaration**.
    - Real-time input validation was implemented to ensure correct data entry.
    - Dynamic addition/removal of fields for **education** and **work experience** details.
2. **Backend (Node.js and Express.js):**
    - RESTful APIs to handle form submission and data retrieval.
    - Integrated **Joi validation** to enforce server-side input constraints.
    - Managed file uploads (e.g., passport photos, mark sheets, and Ph.D. documents) using **Multer** middleware.
    - Stored data in **MongoDB** in a structured format.
3. **Database (MongoDB):**
    - Designed a schema-based storage system to maintain all applicant data.
    - Managed collections for storing candidate details such as:
        - **Personal Information**
        - **Educational Details**
        - **Work Experience**
        - **Publications**
        - **Uploaded Documents**
4. **Data Processing and Filtering:**
    - Shortlisting of candidates based on custom filters like **qualification**, **discipline**, and **work experience**.

- Admin functionalities for viewing, filtering, and managing the list of applicants.

5. **Reporting:**
    - Generated **Excel** and **Word** output for shortlisted candidates.
    - Facilitated ease of review for selection committees.

Appendix B: Tools and Technologies Used

The following technologies were employed for the successful implementation of the project:

Component	**Technology**
Frontend	React.js
Backend	Node.js, Express.js
Database	MongoDB
File Upload Handling	Multer
Data Validation	Joi
CSS and Styling	Plain CSS
Output Formats	Excel and Word files

Appendix C: Code Organization

The project directory is organized as follows:

```
DRDO_Application_Project/
│
├── frontend/
│   ├── src/
│   │   ├── App.js              # Main React Component
│   │   ├── PersonalInfo.js     # Personal Information Form
│   │   ├── Qualifications.js   # Qualifications Form
│   │   ├── DocumentsUpload.js  # Document Upload and Declaration
│   │   ├── ReviewApply.js      # Final Review and Submission
│   │   ├── App.css             # Styling for Components
│   │
│   └── public/
│       ├── logo1.png           # DRDO Logo
│       ├── logo2.png           # Secondary Logo
│
├── backend/
│   ├── server.js               # Backend Server with API Endpoints
│   ├── models/
│   │   ├── FormSchema.js       # MongoDB Schema
│   │
│   ├── uploads/                # Uploaded Files Storage
│
└── package.json                # Project Dependencies
```

Appendix D: Sample Input and Output

1. **Sample Input (Form Data):**

```
{
  "applicationFor": "JRF(DRDO Fellowship)",
  "fullName": "John Doe",
  "fatherName": "Mr. Doe",
  "gender": "Male",
  "dateOfBirth": "1998-05-12",
  "highestQualification": "M. Tech.",
  "discipline": "Physics",
  "contactNumbers": "1234567890",
  "educationDetails": [
    { "degree": "B.Tech", "board": "IIT Delhi", "year": 2020, "percentage": 87.5 }
  ],
  "workExperienceDetails": [
    { "post": "Research Intern", "company": "DRDO", "location": "Delhi",
      "from": "2021-06", "to": "2022-06" }
  ]
}
Project Dependencies
```

2. **Sample Output (Shortlisted Candidates in Excel):**

Name	Qualification	Discipline	Experience
John Doe	M. Tech	Physics	Research Intern
Tarun Kumar	PhD	Electronics	Postdoc Fellow

CHAPTER 10

REFERENCES

The sources, documents, and materials consulted during the planning, design, and implementation phases of the project are

1. Books and Publications:
    - Smith, J. (2021). *Web Development with MERN Stack*. TechPress.
    - Johnson, R. (2020). *Database Design and Development: MongoDB in Action*. DataWorks Publishing.
2. Websites and Online Resources:
    - DRDO Official Website: https://www.drdo.gov.in/drdo/labs-andestablishments/solid-state-physics-laboratory-sspl
    - MongoDB Documentation: https://docs.mongodb.com/
    - React Official Documentation: https://legacy.reactjs.org/docs/gettingstarted.html
    - Express.js Guide: https://expressjs.com/en/guide/routing.html
    - Node.js Documentation: https://nodejs.org/docs/