RAJALAKSHMI ENGINEERING COLLEGE

RAJALAKSHMI NAGAR, THANDALAM – 602 105

RAJALAKSHMI
ENGINEERING COLLEGE
An AUTONOMOUS Institution
Affiliated to ANNA UNIVERSITY, Chennai

CS19442 SOFTWARE ENGINEERING CONCEPTS LAB

Laboratory Record
Note Book

Name :…… ……...

Year / Branch / Section : …….. …….

University Register No. : …….. ……

College Roll No. : ……

Semester : ……

Academic Year : …….

RAJALAKSHMI ENGINEERING COLLEGE

RAJALAKSHMI NAGAR, THANDALAM – 602 105

BONAFIDE CERTIFICATE

Name:_____

Academic Year: _____ Semester:____Branch:_____

Register No: | |

Certified that this is the bonafide record of work done by the above student in the CS19442-Software Engineering Concepts Laboratory during the year 2023- 2024

Signature of Faculty-in-charge

Submitted for the Practical Examination held on _____

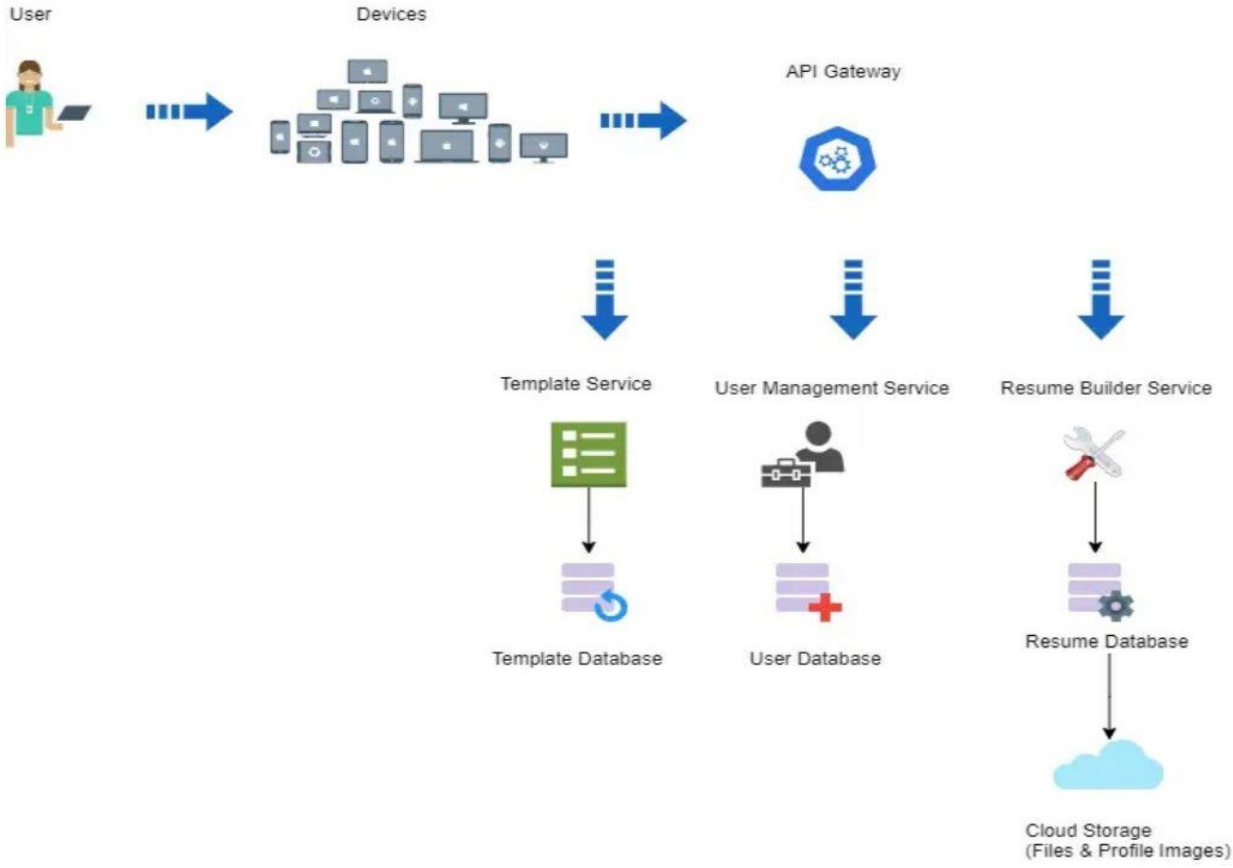
Internal Examiner External Examiner

INDEX

OVERVIEW OF THE PROJECT:

The "Resume Builder" web application is designed to streamline the process of creating professional resumes. In today's competitive job market, a well-crafted resume is crucial for securing job opportunities. Many job seekers struggle to create effective resumes due to lack of design skills or understanding of what employers seek, often leading to missed job opportunities. Data shows that candidates with poorly formatted or incomplete resumes are frequently rejected, even if they have the necessary qualifications. Our Resume Builder addresses these issues by offering user-friendly templates and customization options to create standout resumes. Through our application, users can easily create resumes that highlight their strengths and accomplishments, increasing their chances of being noticed by potential employers.

DEPLOYMENT DIAGRAM:



SOFTWARE REQUIREMENTS SPECIFICATION

EX.NO.1 **DATE: 20-02-2024**

CONTENTS

RESUME BUILDER

1. Introduction

The Resume Builder software aims to provide users with a platform to create professional resumes effortlessly. This document outlines the requirements for developing the Resume Builder application, including its features, functionality, and constraints.

1.1. Purpose

The purpose of a resume builder project is to provide

individuals with a user-friendly tool to create professional resumes quickly and efficiently. These projects typically involve creating a web application where users can input their personal information, educational background, work

experience, skills, and any other relevant details. The resume builder then formats this information into a well-structured and visually appealing resume template

2. Scope

The Resume Builder software will enable users to create, edit, and manage their resumes efficiently. It will offer a user-friendly interface with various templates and customization options to cater to different job positions and industries. The application will also incorporate features for exporting resumes in different formats such as PDF, Word, or plain text.

3. Functional Requirements

3.1. User Authentication

• The system shall provide user authentication to ensure secure access to the application.

• Users shall be able to register for an account with a unique username and password.

• Registered users shall be able to log in securely using their credentials.

3.2. Resume Creation

• The system shall allow users to create new resumes from scratch.

• Users shall have the option to choose from a variety of pre designed templates.

• Users shall be able to enter their personal information, including name, contact details, and career objectives. • The application shall provide sections for users to input their education history, work experience, skills, certifications, and other relevant information.

• Users shall have the ability to customize the layout, font styles, and colors of their resumes.

3.3. Resume Editing and Management

• Users shall be able to edit existing resumes to update information or make modifications.

• The system shall allow users to preview their resumes before finalizing changes.

• Users shall have the option to save multiple versions of their resumes for different job applications.

• The application shall support the organization of resumes into categories or folders for easy management.

3.4. Export and Sharing

• The system shall provide options for users to export their resumes in various formats such as PDF, Word, or plain text.

• Users shall be able to download or email their resumes directly from the application.

• The application shall allow users to share their resumes via social media platforms or professional networking sites.

3.5. Search and Filtering

• The system shall include search and filtering functionality to help users find specific resumes or templates quickly.

• Users shall be able to filter resumes based on criteria such as job title, date created, or keywords.

3.6. Help and Support

• The application shall provide user guides or tutorials to assist users in creating effective resumes.

• Users shall have access to customer support services for any technical assistance or inquiries.

4. Non-functional Requirements

4.1. Performance

• The system shall be responsive and provide quick loading times, even with large amounts of data.

• The application shall support simultaneous users without significant degradation in performance.

4.2. Security

• User data shall be encrypted and stored securely to prevent unauthorized access or data breaches.

• The system shall implement measures to protect against common security threats.

4.3. Usability

• The user interface shall be intuitive and easy to navigate, catering to users with varying levels of technical expertise.

• The application shall provide tooltips and contextual help to guide users through the resume creation process.

4.4. Compatibility

• The system shall be compatible with popular web browsers such as Chrome, Firefox, and Safari.

• The application shall be responsive and adapt to different screen sizes, including desktops, tablets, and mobile devices.

5. Constraints:

• The Resume Builder application shall comply with relevant data protection regulations, such as CCPA, to ensure the privacy and security of user data.

• The system shall rely on third-party APIs or libraries for certain functionalities, such as exporting resumes to different file formats.

6. Conclusion:

In conclusion, the Resume Builder project outlined in this Software Requirements Specification (SRS) aims to provide users with a robust and user-friendly platform for creating professional resumes.

SCRUM METHODOLOGY

EX.NO.2 **DATE: 01-03-2024**

1. The Project Vision

Vision Statement: To create a user-friendly resume builder that allows users to create professional resumes quickly and easily.

Goals:

- User-friendly interface
- Multiple resume templates
- Integration with Linked In
- Downloadable formats (PDF, Word)
- Real-time preview

2. The Product Backlog

The product backlog is a prioritized list of features, enhancements, and bug fixes required for the project.

Product Backlog Items:

- User Registration and Authentication
- Profile Creation and Management
- Resume Template Selection
- Real-time Resume Preview
- Resume Editing Tools (Text, Formatting, Sections)
- Export Options (PDF, Word)
- Integration with Linked In
- User Feedback and Rating System
- Admin Panel for Managing Templates

3. The Scrum Team

- **Product Owner:** Responsible for defining the features of the product and prioritizing the backlog.

- **Scrum Master:** Ensures the Scrum process is followed, removes impediments, and facilitates meetings.

- **Development Team:** Cross-functional team responsible for delivering potentially shippable increments at the end of each sprint (includes developers, designers, QA, etc.).

4. Planning the Sprints

- **Sprint Duration:** Typically 2-4 weeks.

- **Sprint Planning Meeting:** The team selects items from the product backlog to commit to during the sprint.

5. Sprint Planning Meeting

- **Goal:** Define what will be delivered in the sprint and how it will be achieved.

- **Input:** Product backlog, team capacity, past performance.

- **Output:** Sprint backlog (tasks for the sprint), sprint goal.

6. Daily Stand-up Meetings

- **Duration:** 15 minutes

- **Purpose:** Discuss what was done yesterday, what will be done today, and identify any impediments.

7. Sprint Execution

- **Development:** Team works on the tasks in the sprint backlog.

- **Testing:** Continuous integration and testing of features.

8. Sprint Review

- **Purpose:** Demonstrate the working product increment to stakeholders.

- **Activities:** Team shows what was accomplished during the sprint. Stakeholders provide feedback.

9. Sprint Retrospective

- **Purpose:** Reflect on the sprint and identify improvements for future sprints.
- **Activities:** Discuss what went well, what didn't, and how to improve.

10. Release Planning

- **Release Goal:** Determine when and what features will be released to the users.
- **Activities:** Prioritize features, finalize the release date, prepare for deployment.

Sprint Breakdown:

Sprint 1: Basic Framework and User Authentication

- Task 1: Set up the project repository and initial project structure.
- Task 2: Implement user registration and authentication.
- Task 3: Create a basic user profile page.

Sprint 2: Resume Template Selection and Profile Management

- Task 1: Develop the template selection feature.
- Task 2: Enhance the user profile with additional details.
- Task 3: Implement basic profile management (edit, update).

Sprint 3: Real-time Preview and Editing Tools

- Task 1: Implement real-time resume preview.
- Task 2: Develop editing tools for text and formatting.
- Task 3: Add different resume sections (work experience, education, skills).

Sprint 4: Export Options and LinkedIn Integration

- Task 1: Implement export to PDF feature.
- Task 2: Develop export to Word feature.
- Task 3: Integrate with LinkedIn to import user data.

Sprint 5: User Feedback and Admin Panel

- Task 1: Create a user feedback system.
- Task 2: Develop an admin panel for managing templates.
- Task 3: Perform final testing and bug fixing.

USER STORIES

EX.NO.3 **DATE:12-03-2024**

1. **Login/Sign up Pages**

User Story 1

As a new user, I **want to** create an account with a unique username and password **so that** I can access the Resume Builder application.

Acceptance Criteria:

- The application provides a sign-up form requiring a unique username and password.
- Users receive a confirmation email upon successful registration.

User Story 2

As a returning user, I **want to** log in securely using my credentials **so that** I can update my resume or access saved templates.

Acceptance Criteria:

- The application provides a login form requiring a username and password.
- Users are redirected to their dashboard upon successful login.

2. **Recommended Templates Feature**

User Story 1

As a student, **I want to** browse and select from a variety of resume templates tailored to entry-level positions **so that** I can find a template that best suits my needs.

Acceptance Criteria:

- The application provides a category of templates specifically for entry-level positions.
- Students can preview templates before selecting one.

User Story 2

As a job seeker, **I want** access to professionally designed templates that highlight my skills and experiences effectively **so that** I can create a compelling resume.

Acceptance Criteria:

- The application includes professionally designed templates suitable for various industries.
- Templates are designed to highlight skills and experiences prominently.

User Story 3

As a researcher, **I want** templates that showcase my academic achievements and research projects prominently **so that** I can present my work effectively.

Acceptance Criteria:

- The application offers templates tailored for academic and research positions.
- Templates include sections for academic achievements and research projects.

3. **Efficient Document Sharing**

User Story 1

As a user, **I want** to share my resume with potential employers in PDF format **so that** would be for easy viewing and printing.

Acceptance Criteria:

- Users can export their resumes to PDF format.

- PDF resumes retain the formatting and design of the selected template.

User Story 2

As a user, **I want** the option to email my resume directly from the application to recruiters or hiring managers **so that** I can streamline the application process.

Acceptance Criteria:

-The application includes a feature to email resumes directly to specified email addresses.

- Users receive a confirmation when the email is sent successfully.

User Story 3

As a user, I **want** to share my resume on Linked In or other professional networks **so that** I can expand my job search network.

Acceptance Criteria:

- Users can share their resumes directly to LinkedIn and other professional networks.

- The shared resumes maintain their formatting and are viewable on the respective platforms.

4. **Update Resume**

User Story 1

As a user, I **want to** edit my existing resume **so that** I can update my new experiences or qualifications.

Acceptance Criteria:

- Users can easily update and edit existing resume entries.

- Changes are saved automatically or upon user command.

User Story 2

As a user, I **want to** add new sections to my resume, such as certifications or volunteer work **so that** enhances my profile.

Acceptance Criteria:

- Users can add new sections to their resumes.

-The application provides templates for various sections like certifications, volunteer work, etc.

5. **Download Resume**

User Story 1

As a user, I **want to** download my resume in Word format **so that** I can tailor it for specific job applications.

Acceptance Criteria:

- Users can export their resumes to Word format.

- Exported resumes maintain the formatting and design of the selected template.

User Story 2

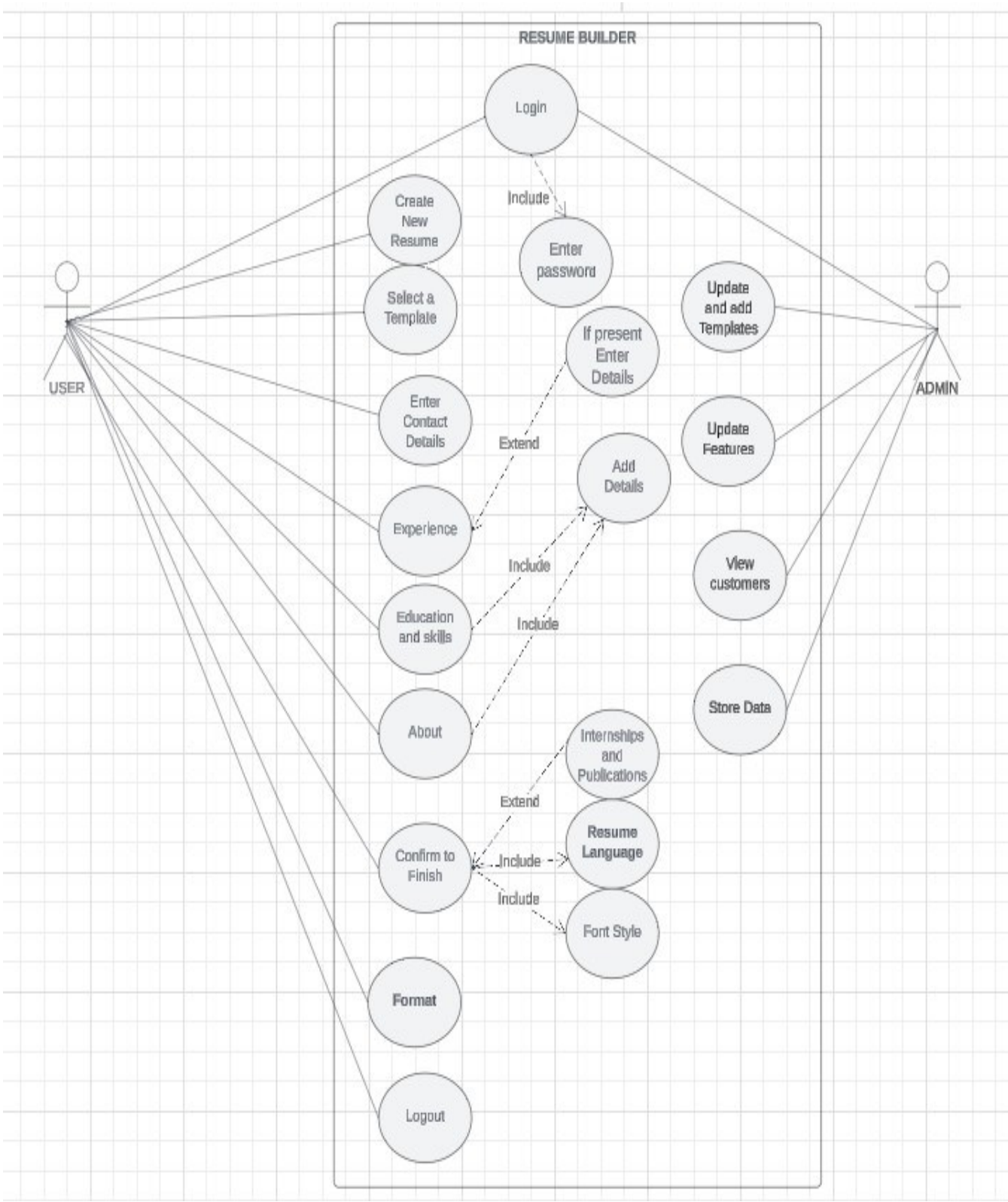
As a user, I **want** the option to save multiple versions of my resume **so that** can be applied for different industries or job roles.

Acceptance Criteria:

- Users can save and manage multiple versions of their resumes within the application.
- Each version can be edited, downloaded, or shared independently.

USE CASE DIAGRAM

EX.NO:4 **DATE:19-03-2024**



USER refers to Student, Researcher, Job seeker.

The use case diagram illustrates the functional interaction between users and the Resume Builder system. It primarily involves two actors: the User and the Admin.

For the User:

Login: The user logs into the system, which includes entering a password. If details are already present, the system retrieves them.

Create New Resume: The user can initiate creating a new resume.

Select a Template: Users can choose from various templates.

Enter Contact Details: Users input their contact information.

Experience: Users add their professional experience.

Education and Skills: Users list their educational background and skills.

About: Users can write a personal statement or summary.

Add Details: This includes additional sections like internships, publications, resume language, and font style.

Confirm to Finish: Users confirm the completion of their resume.

Format: Users can format the final resume.

Logout: Users can log out of the system.

For the Admin:

Update and Add Templates: Admins can manage the resume templates.

Update Features: Admins can update system features.

View Customers: Admins can view user data.

Store Data: Admins ensure that user data is securely stored.

The use case diagram emphasizes the system's capabilities to facilitate seamless resume creation and management. It outlines the step-by-step process users follow, starting from logging in to finalizing their resume. It also showcases the admin's responsibilities in maintaining the system's functionality and ensuring an up-to-date user experience. The interactions and dependencies among different functionalities are depicted clearly, ensuring a coherent flow and understanding of the system's operations.

Non-Functional Requirement(NFR)

EX.NO:5 **DATE:29-03-2024**

1. Performance

- Response Time: The resume builder application shall ensure that all major interactions, such as loading templates, saving resumes, and generating previews, have a response time of less than 2 seconds. This is crucial to provide a seamless and efficient user experience, even when users upload large amounts of data, such as detailed resumes with multiple sections and attachments.

2. Security

- Data Encryption: All user data, including personal information, resume content, and login credentials, shall be encrypted both in transit and at rest using industry-standard encryption algorithms such as AES-256. This will prevent unauthorized access and ensure that sensitive information is protected against data breaches.

3. Usability

-User Interface: The resume builder application shall have an intuitive and easy-to-navigate user interface designed to cater to users with varying levels of technical expertise. This includes clear navigation menus, drag-and-drop functionality for editing resumes, and a user-friendly design for selecting and customizing resume templates. The application shall provide real-time feedback and validation messages to guide users through the resume creation process.

4. Compatibility

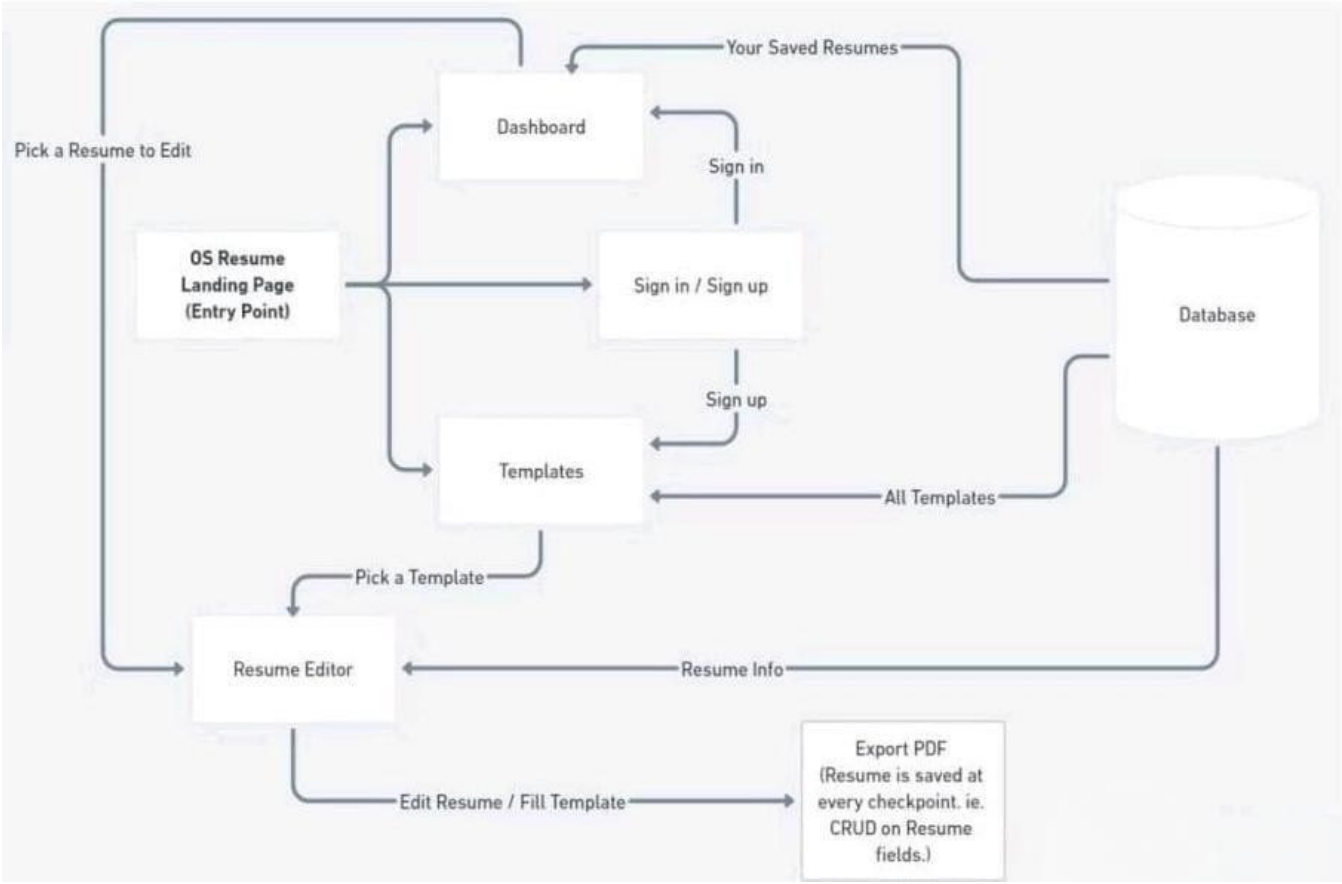
- Browser Compatibility: The resume builder application shall be compatible with the latest versions of popular web browsers, including Chrome, Firefox, Safari, and Edge. This ensures that users can access the application from their preferred browsers without any issues, providing a consistent and reliable user experience.

5. Reliability

- Availability: The resume builder application shall maintain an uptime of 99.9% to ensure high availability for users creating or updating their resumes. This is essential for job seekers who need access to the application at any time, especially when applying for jobs with tight deadlines. Regular maintenance and updates shall be scheduled during off-peak hours to minimize disruptions.

OVERALL PROJECT ARCHITECTURE

EX.NO:6 **DATE:09-04-2024**



The overall architecture for the online resume builder is designed to streamline the process of creating professional resumes for job seekers. The system starts at the OS Resume Landing Page, which serves as the entry point. From here, users can sign in or sign up to access the dashboard, where they can manage their saved resumes.

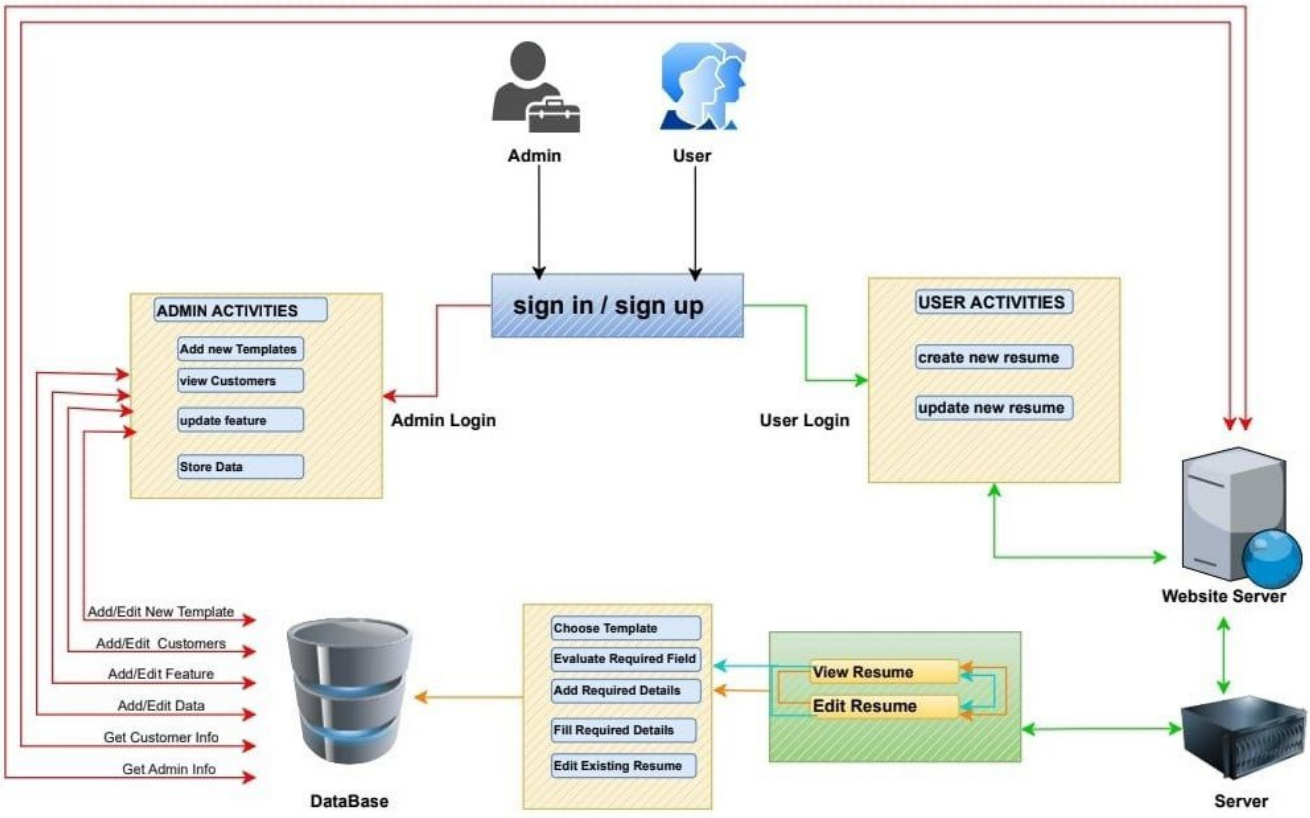
The sign-in/sign-up process is connected to a central database that stores user information and resumes. Once logged in, users can choose from various resume templates, which are also fetched from the database. These templates provide predefined structures that users can fill with their personal, educational, and professional details.

Users pick a template and proceed to the Resume Editor, where they can edit and fill in the chosen template. This editor continuously saves the resume information at every checkpoint, ensuring data integrity and reducing the risk of data loss. Once the resume is complete, users can export it as a PDF.

The architecture is designed to be user-friendly, enabling quick access and management of resumes. The database supports CRUD (Create, Read, Update, Delete) operations on resume fields, ensuring users can efficiently create, modify, and save their resumes. This systematic approach helps job seekers produce well-structured resumes with minimal effort.

BUSINESS ARCHITECTURE

EX.NO:7 **DATE:19-04-2024**



Business Architecture of Resume Builder

Overview

This architecture diagram represents the business logic of a Resume Builder application, highlighting the interactions between different user roles, activities, and system components

Components and Workflow

Users and Roles

- **Admin**: Manages templates, customer information, and other administrative features.
- **User**: Creates and updates resumes.

Admin Activities

- **Add new Templates**: Admins can add new resume templates.
- **View Customers**: Admins can view customer information.
- **Update Feature**: Admins can update features of the resume builder.
- **Store Data**: Admins can store data related to templates and customer information.

These activities interact with the Database to add or edit templates, customers, features, and retrieve customer and admin information.

User Activities

- **Create New Resume**: Users can create new resumes.
- **Update New Resume**: Users can update existing resumes.
- These activities involve selecting a template, evaluating required fields, and filling in necessary details, which are facilitated by the Website Server and stored in the Database**.**

Resume Management

- **Choose Template**: Users select a resume template from available options.
- **Evaluate Required Field**: Users review and complete necessary fields for the resume.
- **Fill Required Details**: Users enter the details into the selected template.
- **Edit Existing Resume**: Users can edit previously created resumes.

These processes interact with the Database to store and retrieve resume information.

Server Interactions

- **Website Server**: Handles requests and responses between the user interface and the backend services.
- **Main Server**: Processes the data and interactions, ensuring smooth functionality of the application.

Data Flow

Admins and Users authenticate through the sign-in/sign-up interface.

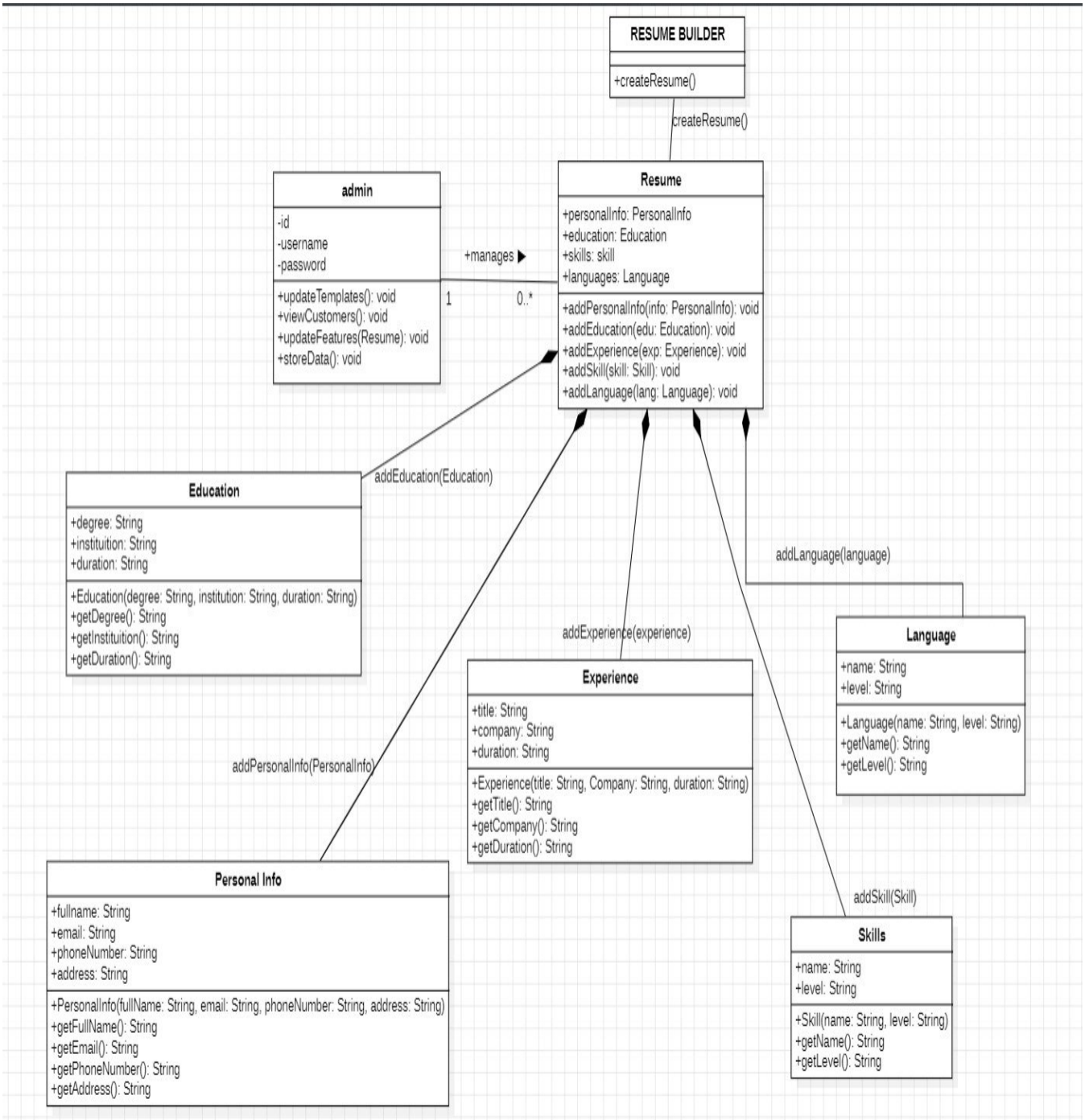
Admin activities interact directly with the database to manage templates, customer information, and other features.

User activities primarily interact with the website server to create and update resumes, which then interact with the main server and the database for data processing and storage.

The website server manages the front-end interactions and communicates with the main server for backend processing.

CLASS DIAGRAM

EX.NO:8 **DATE:30-04-2024**



Class Diagram Overview:

The class diagram depicts the structure of the Resume Builder application, showcasing the classes involved, their attributes, methods, and relationships. It represents how different components of the system interact to create and manage resumes.

Explanation of Class Components:

1. **ResumeBuilder**
   - Methods: createResume()
   - Purpose: Main system class responsible for initiating resume creation.

2. **Resume**
   - Attributes: personalInfo, education, skills, languages
   - Methods:
     - addPersonalInfo(PersonalInfo info): Adds personal information.
     - addEducation(Education edu): Adds educational details.
     - addExperience(Experience exp): Adds work experience.
     - addSkill(Skill skill): Adds skills.
     - addLanguage(Language lang): Adds languages.
   - Purpose: Represents the resume, containing personal info, education, skills, experience, and languages.

3. **Admin**
   - Attributes: id, username, password
   - Methods:
     - updateTemplates(): Updates resume templates.
     - viewCustomers(): Views customer details.
     - updateFeatures(Resume): Updates features of a resume.
     - storeData(): Stores data in the database.
   - Purpose: Manages the system and handles administrative tasks.

4. **PersonalInfo**

- Attributes: fullName, email, phoneNumber, address
- Methods:
  - PersonalInfo(String fullName, String email, String phoneNumber, String address): Constructor.
  - Getters for each attribute.
- Purpose: Stores personal information of the user.

5. **Education**

- Attributes: degree, institution, duration
- Methods:
  - Education(String degree, String institution, String duration): Constructor.
  - Getters for each attribute.
- Purpose: Stores educational background.

6. **Experience**

- Attributes: title, company, duration
- Methods:
  - Experience(String title, String company, String duration): Constructor.
  - Getters for each attribute.
- Purpose: Stores work experience details.

7. **Skills**

- Attributes: name, level
- Methods:
  - Skill(String name, String level): Constructor.
  - Getters for each attribute.
- Purpose: Stores skills information.

8. **Language**

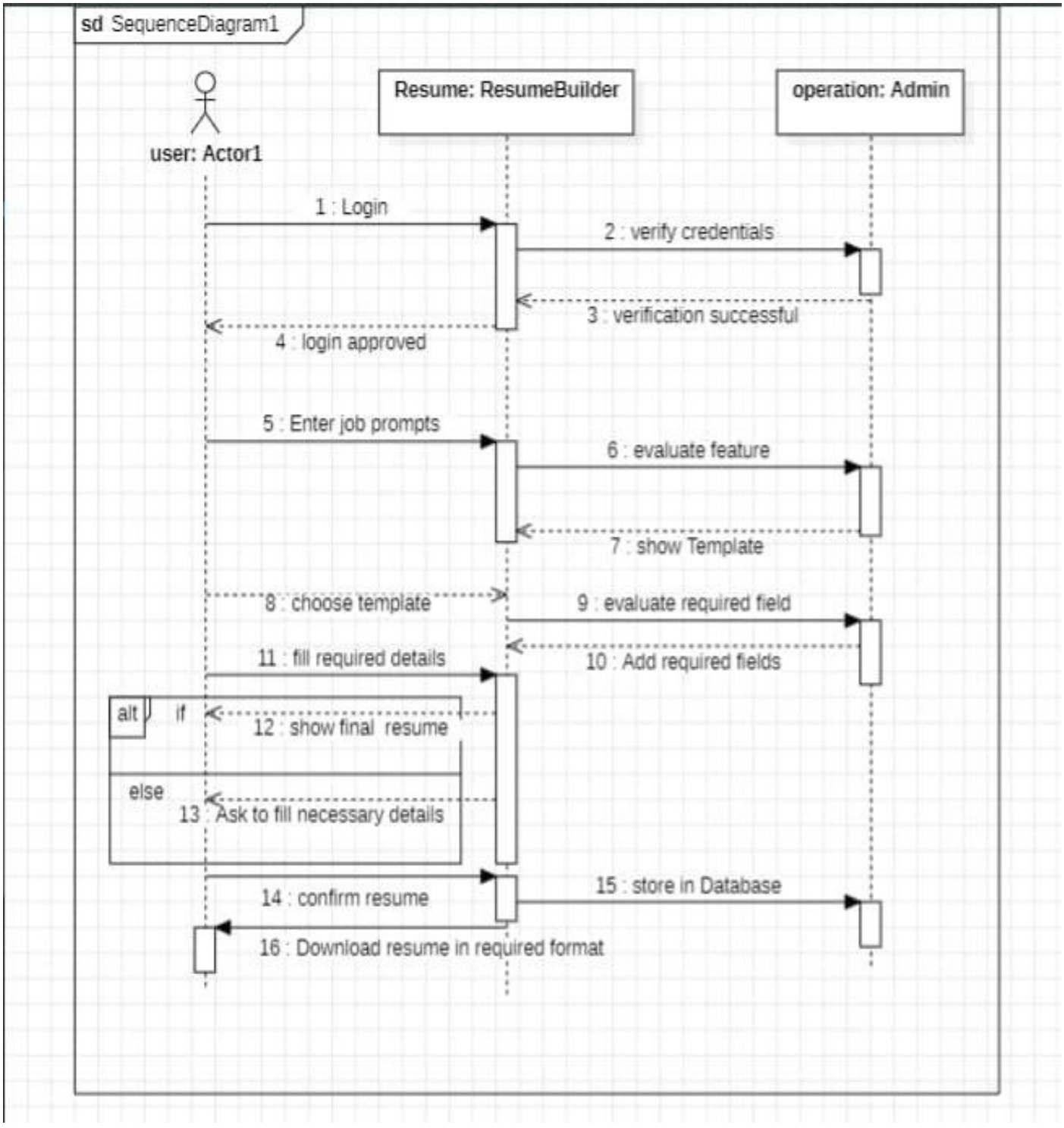
- Attributes: name, level
- Methods:
  - Language(String name, String level): Constructor.
  - Getters for each attribute.
- Purpose: Stores language proficiency details.

Relationships

- Admin to Resume: Admin can manage multiple resumes.
- Resume to Other Classes Resume has associations with PersonalInfo, Education, Experience, Skills, and Language to aggregate different resume sections

SEQUENCE DIAGRAM

EX.NO:9 **DATE:10-05-2024**



Sequence Diagram Overview:

This sequence diagram illustrates the process flow for a resume-building application, detailing how users interact with the system to create and download a resume.

Actors

1. User: The person using the resume builder.
2. Admin: Handles credential verification.

Sequence Flow

1. User Login:

- The user logs in.
- The system sends credentials to the Admin.
- Admin verifies and approves the login.

2. **Input Job Prompts:**

- User enters job prompts into the system.
- The system evaluates and shows a suitable template.

3. **Template Selection and Details:**

- User selects a template.
- The system identifies required fields and prompts the user to fill them.

4. **Resume Finalization:**

- If all required details are filled, the system shows the final resume.
- If details are missing, the system asks the user to complete them.

5. **Confirmation and Storage:**

- User confirms the resume.
- The system stores the resume in the database.

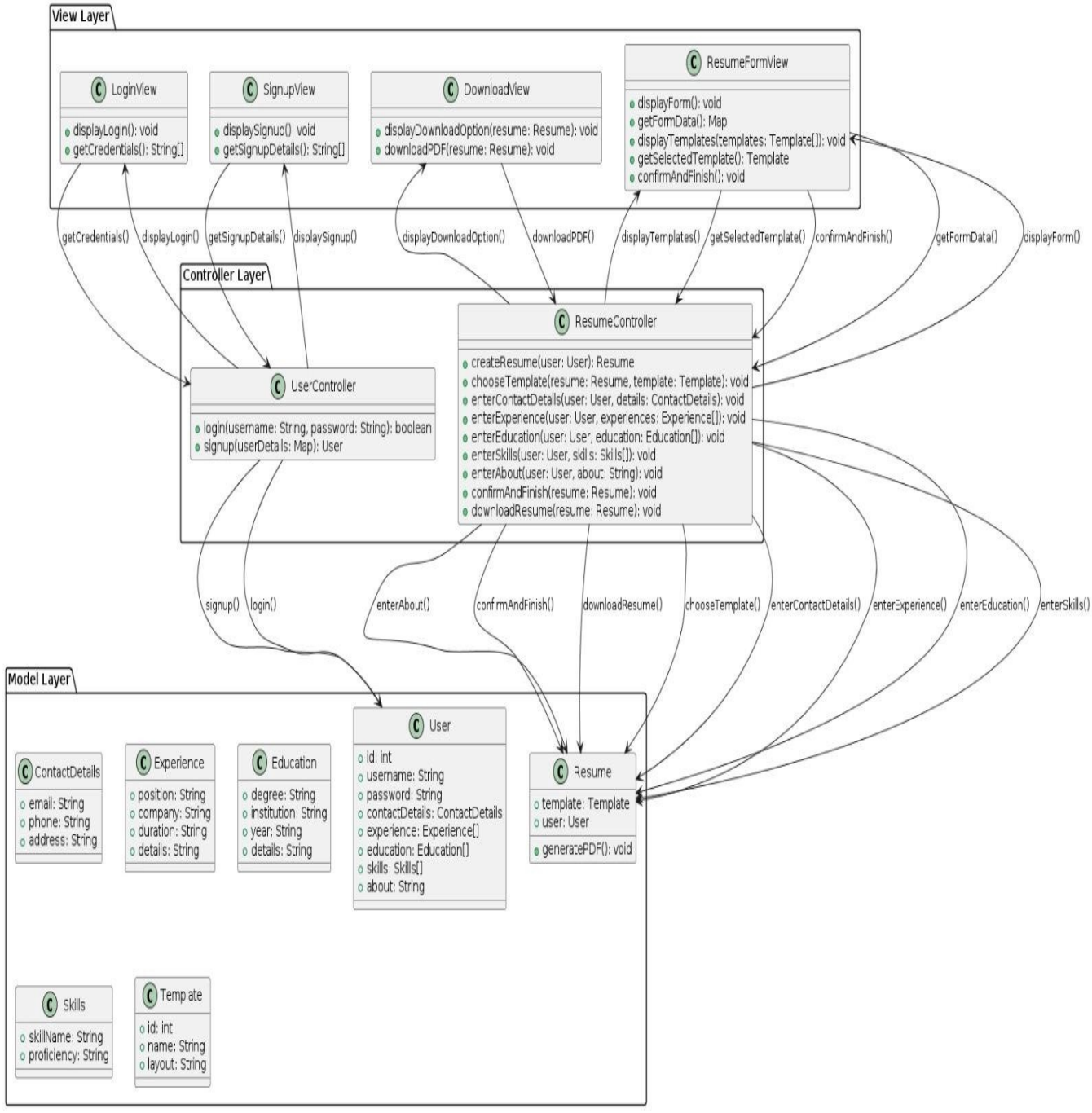
6. **Download Resume:**

- User downloads the resume in the desired format.

ARCHITECTURAL PATTERNS

EX.NO.10 DATE:17-05-2024

MODEL VIEW CONTROLLER ARCHITECTURE:



Model Layer

The Model layer consists of several classes that represent the data and business logic of the application:

1. **User**: Contains user details like **id**, **username**, **password**, and arrays of **ContactDetails**, **Experience**, **Education**, and **Skills**.
2. **ContactDetails**: Represents contact information with fields for **email**, **phone**, **address**, and **LinkedIn**.
3. **Experience**: Contains details of work experience such as **position**, **company**, **duration**, and **details**.
4. **Education**: Stores educational qualifications with fields like **degree**, **institution**, **year**, and **details**.
5. **Skills**: Represents skills with fields for **skillName** and **proficiency**.
6. **Template**: Contains information about resume templates with fields for **name**, **description**, and **layout**.
7. **Resume**: Integrates all aspects of a resume, holding references to **Template** and **User** objects, and a method to **generatePDF()**.

View Layer

The View layer is responsible for displaying the user interface and receiving user input:

1. **LoginView**: Displays the login interface and retrieves credentials from the user.
    - displayLogin()
    - getCredentials()
2. **SignupView**: Shows the signup interface and collects user registration details.
    - displaySignup()
    - getSignupDetails()

3. **DownloadView**: Manages the download options for the resume.
    - displayDownloadOption()
    - downloadPDF()
4. **ResumeFormView**: Displays the form for entering resume details, selects templates, and confirms the resume.
    - displayForm()
    - getFormData()
    - displayTemplateOptions()
    - getSelectedTemplate()
    - confirmAndFinish()

Controller Layer

The Controller layer acts as an intermediary between the Model and View layers, handling the application logic and user input:

1. **UserController**: Manages user authentication and registration.
    - login()
    - signup()

2. **ResumeController**: Handles operations related to resume creation and modification.
    - createResume()
    - chooseTemplate()
    - enterContactDetails()
    - enterExperience()
    - enterEducation()
    - enterSkills()
    - enterAbout()
    - confirmAndFinish()
    - downloadResume()

Interactions between Layers

The diagram shows the interactions between the layers:

1. **Login and Signup**:
    - **LoginView** calls **getCredentials()** and passes them to **UserController**'s **login()**.
    - **SignupView** calls **getSignupDetails()** and passes the details to **UserController**'s **signup()**.
2. **Resume Creation**:
    - **ResumeFormView** displays forms and templates, calling methods like **displayForm()**, **displayTemplateOptions()**, and **getSelectedTemplate()**.
    - The data collected is processed by **ResumeController** through methods such as **createResume()**, **chooseTemplate()**,

enterContactDetails(), **enterExperience()**, **enterEducation()**, **enterSkills()**, **enterAbout()**, and **confirmAndFinish()**.

3. **Resume Download**:
    - **DownloadView** handles the display and download of the resume through **displayDownloadOption()** and **downloadPDF()**.
    - **ResumeController**'s **downloadResume()** facilitates the actual download process.

Example Workflow

1. **User Login**:
    - User inputs credentials in **LoginView**.
    - **UserController** validates the credentials and allows access if correct.
2. **Resume Creation**:
    - User navigates to **ResumeFormView** and starts entering resume details.
    - Details are passed to **ResumeController**, which updates the **Resume** model with the user-provided data.
    - User selects a template and confirms the resume.
3. **Resume Download**:
    - User navigates to **DownloadView** to download the resume.
    - **DownloadView** invokes **ResumeController** to generate and download the PDF version of the resume.

In this architecture, the separation of concerns ensures that the application is modular, maintainable, and scalable, with each layer having a distinct responsibility.

TEST CASES

Test Case 1: Creating a New Resume

Description: Verify that a user can create a new resume from scratch.

Preconditions:

- User is logged into the Resume Builder application.

Test Steps:

1. Navigate to the "Create New Resume" section.
2. Enter basic personal information (Name, Address, Contact Information).
3. Add educational qualifications (School/College name, Degree, Year of passing).
4. Add work experience (Company name, Position, Duration).
5. Add skills and certifications.
6. Click the "Save" button.

Expected Results:

- The application should save the entered information.
- The user should see a confirmation message indicating the resume has been saved successfully.
- The newly created resume should appear in the list of available resumes for the user.

Test Case 2: Editing an Existing Resume

Description: Verify that a user can edit an existing resume.

Preconditions:

- User is logged into the Resume Builder application.
- User has at least one saved resume.

Test Steps:

1. Navigate to the "My Resumes" section.
2. Select an existing resume to edit.
3. Modify the personal information (e.g., update phone number).
4. Modify work experience (e.g., add a new job position).
5. Click the "Save" button.

Expected Results:

- The application should update the resume with the new information.
- The user should see a confirmation message indicating the resume has been updated successfully.
- The modified resume should reflect the changes made.

Test Case 3: Downloading a Resume

Description: Verify that a user can download a resume in PDF format.

Preconditions:

- User is logged into the Resume Builder application.
- User has at least one saved resume.

Test Steps:

1. Navigate to the "My Resumes" section.
2. Select an existing resume to download.
3. Click the "Download" button.
4. Choose "PDF" as the format.

Expected Results:

- The application should generate a PDF file of the resume.
- The user should see a prompt to download the PDF file.
- The downloaded PDF should contain the correct and formatted information of the resume.

Test Case 4: Using a Template for Resume Creation

Description: Verify that a user can create a resume using a predefined template.

Preconditions:

- User is logged into the Resume Builder application.

Test Steps:

1. Navigate to the "Create New Resume" section.
2. Select a predefined template from the available options.
3. Enter personal information, educational qualifications, work experience, skills, and certifications.
4. Click the "Preview" button to see the resume with the selected template.
5. Click the "Save" button.

Expected Results:

- The application should save the entered information with the selected template.
- The user should see a confirmation message indicating the resume has been saved successfully.
- The preview should display the resume with the correct template formatting.
- The newly created resume should appear in the list of available resumes for the user with the applied template.

Test Case 5: Deleting a Resume

Description: Verify that a user can delete an existing resume.

Preconditions:

- User is logged into the Resume Builder application.
- User has at least one saved resume.

Test Steps:

1. Navigate to the "My Resumes" section.
2. Select a resume to delete.
3. Click the "Delete" button.
4. Confirm the deletion in the confirmation dialog.

Expected Results:

- The application should delete the selected resume.
- The user should see a confirmation message indicating the resume has been deleted successfully.
- The deleted resume should no longer appear in the list of available resumes for the user.

These test cases cover the fundamental functionalities of a Resume Builder application, ensuring that users can create, edit, download, use templates, and delete resumes effectively.