Intellectual Property Facilitation Centre Management System

Software Requirements Specification

INT221

MVC Programming

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Reg. No. | Roll No. | Section |
| Aaditya Kumar | 12203627 | 56 | K22LE |
| Aftab Alam | 12303861 | 50 | K22LE |

Prepared for

Continuous Assessment 2

Autumn 2024

**TABLE OF CONTENTS**

[1. Introduction 1](#_Toc182086778)

[1.1 Purpose 1](#_Toc182086780)

[1.2 Scope 1](#_Toc182086781)

[1.3 Definitions, Acronyms, and Abbreviations 1](#_Toc182086782)

[1.4 References 2](#_Toc182086783)

[1.5 Overview 2](#_Toc182086784)

[2. General Description 2](#_Toc182086785)

[2.1 Product Perspective 2](#_Toc182086786)

[2.2 Product Functions 3](#_Toc182086787)

[2.3 User Characteristics 3](#_Toc182086788)

[2.4 General Constraints 3](#_Toc182086789)

[2.5 Assumptions and Dependencies 4](#_Toc182086790)

[3. Specific Requirements 4](#_Toc182086791)

[3.1 External Interface Requirements 4](#_Toc182086792)

[3.2 Functional Requirements 5](#_Toc182086793)

[3.4 Design Constraints 6](#_Toc182086794)

[3.5 Other Requirements 6](#_Toc182086795)

[4. Analysis Models 7](#_Toc182086796)

[4.1 User Flow Diagrams 7](#_Toc182086797)

[4.2 Entity-Relationship (ER) Diagram 9](#_Toc182086798)

[5 Output Screenshots 12](#_Toc182086799)

# 1. Introduction

## This Software Requirement Specification (SRS) document provides a comprehensive overview of the requirements for the "IPR Facilitation Centre." The document is intended to aid software engineers in designing, developing, and implementing the system effectively. This document ensures that all stakeholders have a clear understanding of the system's purpose, functionality, and constraints.

## 1.1 Purpose

The purpose of this document is to define the functional and non-functional requirements of the Intellectual Property Facilitation Centre (IPFC) Management System. This system is intended to streamline and digitize the day-to-day activities of an IP Facilitation Centre, enabling users such as administrators, IP creators, and reviewers to efficiently manage their workflows. The document outlines all necessary specifications required for the successful design, development, testing, and implementation of the system.

The primary stakeholders of this system include the IPFC administrators, staff members, inventors or researchers submitting intellectual property for processing, as well as reviewers and legal experts involved in the evaluation and processing of IP claims. This SRS serves as a mutual agreement between the client and the development team to ensure a clear understanding of the product scope and functionalities.

## 1.2 Scope

The *IPFC Management System* will be a comprehensive web-based platform developed using the Laravel framework. It will provide an integrated interface for managing various aspects of IP facilitation, including but not limited to:

* User authentication and access control
* Submission and tracking of intellectual property applications
* Administrative dashboards for managing submissions, approvals, and status updates
* Communication modules for interaction between applicants and officials
* Report generation and data analytics

By automating and digitizing core operations, the system aims to improve the efficiency, accuracy, and traceability of the intellectual property facilitation process. It will also help reduce the manual workload, eliminate data redundancy, and improve response time for application reviews and status updates.

The system will be scalable and designed with modular architecture, ensuring easy maintenance and upgrades in the future. It will support multiple user roles with varying access levels and be optimized for both desktop and mobile web access.

## 1.3 Definitions, Acronyms, and Abbreviations

* SRS: Software Requirement Specification
* IP – Intellectual Property
* IPFC – Intellectual Property Facilitation Centre
* SRS – Software Requirements Specification
* UI – User Interface
* UX – User Experience
* CRUD – Create, Read, Update, Delete
* MVC – Model View Controller (Laravel architecture)

## 1.4 References

1. IEEE Guide to SRS, 1998 Edition.
2. XAMPP Documentation [https://www.apachefriends.org](https://www.apachefriends.org/).
3. Composer Documentation https://getcomposer.org/doc/.
4. PHP Manual <https://www.php.net/manual/en/>.

## 1.5 Overview

This document is organized into several sections for clarity:

* **Introduction:** Defines the goals, scope, and target audience of the system.
* **Overall Description:** Describes user needs, product perspective, and system environment.
* **Specific Requirements:** Lists detailed functional and non-functional requirements of the system.
* **System Features:** Breaks down the core modules and their functionalities.
* **External Interface Requirements:** Defines user, hardware, software, and communication interfaces.
* **Other Requirements:** Includes security, reliability, compliance, and performance needs.

# 2. General Description

This section outlines the general context in which the **"Intellectual Property Facilitation Centre (IPFC) Management System"** operates. It highlights the system’s interaction with other components, summarizes its major functions, and explains user characteristics, constraints, and assumptions affecting development.

## 2.1 Product Perspective

The **IPFC Management System** is an independent, web-based application built using the Laravel PHP framework. It is developed to streamline the management, tracking, and facilitation of intellectual property-related services across various IP facilitation centers. The system is intended for government bodies, innovation hubs, and educational institutions aiming to manage client requests, services, and IP records efficiently.

**Related Products/Projects:**

* **Database Management Tools** (e.g., MySQL Workbench): Used for storing and retrieving structured data related to clients, IP records, and services.
* **Laravel Framework Utilities**: Utilizes Composer for dependency management and Laravel packages for authentication, routing, and data handling.
* **Web Servers**: Requires Apache or Nginx to serve the application on a local or cloud environment.

## 2.2 Product Functions

The system supports the following primary functions:

1. **User Authentication**: Login and signup functionality for both admin and staff users using Laravel’s built-in authentication system.
2. **Dashboard Interface**: A user-friendly dashboard to manage services, track applications, and view statistics.
3. **Client and Service Management**: Admins can add, edit, view, and delete client details and their respective IP services (patents, trademarks, etc.).
4. **Report Generation**: Generate summary and detailed reports related to IP services and user activity in downloadable PDF formats.
5. **Access Control**: Role-based access restrictions to prevent unauthorized use and modification of sensitive data.
6. **Index Page**: Public-facing landing page with information about the center and login/signup links for new or existing users.

## 2.3 User Characteristics

The system is built to accommodate the following user groups:

1. **Staff Users (Facilitators/Operators)**:
   * + **Technical Expertise**: Basic knowledge of web forms and dashboards.
     + **Responsibilities**: Log in, enter client data, update IP status, and generate client reports.
2. **Admin Users**:
   * **Technical Expertise**: Moderate understanding of web interfaces, IP workflow management, and report generation.
   * **Responsibilities**: Oversee all operations, manage user accounts, and ensure smooth system usage.

All users operate within a secure and controlled digital environment provided by the IP facilitation center.

## 2.4 General Constraints

The system is subject to the following limitations:

1. **Hardware Requirements**:
   * A computer/server capable of running Laravel applications (PHP, MySQL, Composer, etc.).
2. **Software Requirements**:
   * Laravel 10.x or higher with Composer dependency management.
   * Local or remote server setup using Apache/Nginx.
   * PHP version 8.1 or higher.
3. **Security Constraints**:
   * Encrypted password storage and CSRF protection.
   * Restricted access based on roles and user authentication.
4. **Performance Constraints**:
   * Efficient handling of up to 50 concurrent users.
   * Report generation within 5 seconds for datasets up to 1,000 records.

## 2.5 Assumptions and Dependencies

1. **Environment Assumptions**:
   * Laravel is installed with required PHP extensions.
   * The MySQL database is correctly configured and accessible.
2. **Dependency on External Tools**:
   * The application uses **Composer** for dependency management.
   * **Laravel packages** for authentication, PDF generation, and UI enhancements.
3. **Browser Compatibility**:
   * Users will access the system using modern web browsers (e.g., Chrome, Firefox, Edge) supporting HTML5, CSS3, and JavaScript.
4. **Operating System**:
   * Compatible with operating systems capable of running Laravel environments (Windows, Linux, macOS).

**Note**: Changes in assumptions or external dependencies may necessitate corresponding updates to the system’s requirements or implementation strategy.

# 3. Specific Requirements

This section outlines the detailed **functional and non-functional requirements** for the **"Intellectual Property Facilitation Centre (IPFC) Management System."** These requirements guide the design, implementation, and testing of the software.

## 3.1 External Interface Requirements

* + 1. **User Interfaces**
* **Login Page**: A clean and responsive login form for existing users, including email/username and password fields.
* **Signup Page**: Registration form for new users with necessary input fields (name, email, password, role selection).
* **Dashboard**:
  + **For Staff Users**: Interface to add/manage client details, submit service requests, and track progress.
  + **For Admins**: Full access to manage users, oversee IP service requests, and generate reports.
* **Index Page**: A public-facing landing page offering general information about the IPFC, including login/signup navigation.
* **Navigation**: Simple sidebar or top menu navigation for smooth movement between modules.
* **Error Messages**: Clear, user-friendly messages for form validation, login errors, and system issues.

**3.1.2 Hardware Interfaces**

* **Processor**: Dual-core 2.0 GHz or higher
* **RAM**: 4 GB or more
* **Storage**: 10 GB free disk space

**3.1.3 Software Interfaces**

* **Server**: Apache (via XAMPP).
* **Database**: MySQL for data storage and retrieval.
* **Backend**: PHP for server-side processing.
* **Frontend**: HTML, CSS (Bootstrap), and JavaScript for user interaction.
* **External Libraries**: Dependency management using Composer.

**3.1.4 Communications Interfaces**

* **HTTP/HTTPS Protocols**: Used for communication between client and server.
* **Database Communication**: Secure interactions between PHP scripts and the MySQL

database.

## 3.2 Functional Requirements

**3.2.1.1 Introduction**

All users must authenticate with valid credentials. Access is controlled through user roles (Admin, Staff).

**3.2.1.2 Inputs**

* Email/Username
* Password

**3.2.1.3 Processing**

* Validate user credentials against the database using Laravel’s Auth system

**3.2.1.4 Outputs**

* Redirect to respective dashboards upon successful login
* Show error on login failure

**3.2.1.5 Error Handling**

* Account lockout or throttling after 5 failed attempts

**3.2.2 Client and Service Management**

**3.2.2.1 Introduction**  
Staff users input and manage client data, track IP services like patents and trademarks.

**3.2.2.2 Inputs**

* Client name, contact, type of IP service, submission date, progress status

**3.2.2.3 Processing**

* Store client and service records in the database

**3.2.2.4 Outputs**

* Confirmation of successful record entry/update

**3.2.2.5 Error Handling**

* Display appropriate messages for validation errors or database issues

**3.2.3 Report Generation**

**3.2.3.1 Introduction**  
Admins generate reports for client activities, filtered by date range or service type.

**3.2.3.2 Inputs**

* Start date, End date

**3.2.3.3 Processing**

* Query the database to fetch matching service records

**3.2.3.4 Outputs**

* Generate downloadable PDF report using Laravel PDF packages

**3.2.3.5 Error Handling**

* Notify if no data matches the criteria

**3.3 Non-Functional Requirements**

**3.3.1 Performance**

* Support up to 50 simultaneous users without degradation
* PDF generation within 5 seconds for datasets up to 1,000 records

**3.3.2 Reliability**

* Target 99.9% uptime during operational hours

**3.3.3 Availability**

* System should be accessible 24/7 except for scheduled maintenance

**3.3.4 Security**

* Use bcrypt or Argon2 hashing for passwords
* Role-Based Access Control (RBAC) for permissions
* Sanitize inputs to prevent SQL Injection and XSS

**3.3.5 Maintainability**

* Follow Laravel MVC structure with modular, reusable code
* Clear comments and documentation for ease of maintenance

**3.3.6 Portability**

* Compatible with Windows, macOS, and Linux environments using Laravel-supported tools

## 3.4 Design Constraints

* Must comply with organizational security and data protection policies
* Adhere to GDPR or equivalent privacy laws where applicable
* Should run on environments with Laravel and supporting stacks (XAMPP, Laravel Sail, Homestead, etc.)
* Designed to be responsive and browser-compatible (latest versions of Chrome, Firefox, Edge)

## 3.5 Other Requirements

* **Error Logging**: All system-level and application errors must be logged (e.g., Laravel’s storage/logs/) for debugging and auditing
* **Documentation**: Provide detailed user and developer documentation, including setup, API references, and usage guides
* **Localization**: Default language is English. Must allow easy integration of additional languages in future using Laravel localization features

# 4. Analysis Models

List all analysis models used in developing specific requirements previously given in this SRS. Each model should include an introduction and a narrative description. Furthermore, each model should be traceable the SRS’s requirements.

## User Flow Diagrams

For the IPR Facilitation Center Management System, the user flow diagram models the sequential interactions a user undergoes while accessing and managing intellectual property projects. This model provides a high-level overview of how users interact with various components such as login, project creation, and profile access. The model helps ensure clarity of expected behavior and supports traceability of functional requirements, such as user authentication, project submission, and data validation.

1. Start : The user opens the application and logs into the system using valid credentials.
2. Action : After login, the user is presented with the system dashboard and chooses between multiple options:

* "Add New IPR Project"
* "View Existing Projects"
* "View/Edit Profile"
* "Logout"

1. Add New Project Flow

* Action: User selects the "Add New Project" option from the dashboard.
* Action: User fills out the project entry form including fields such as:
  + Title of the IPR project
  + Description of the intellectual property
  + Inventor or applicant details
  + Filing status (e.g., Draft, Submitted, Filed)
  + Submission date (auto-filled or manually selected)
* Decision: The system validates if the entered data is complete and accurate.
  + If any required field is missing or invalid, an error message is displayed and the user can correct the information.
  + If all data is valid, the user proceeds to submit the project.

1. Submit Project

* Action: After successful validation, the user submits the project for review or approval.
* End Point: The system confirms successful submission and stores the project details in the database.

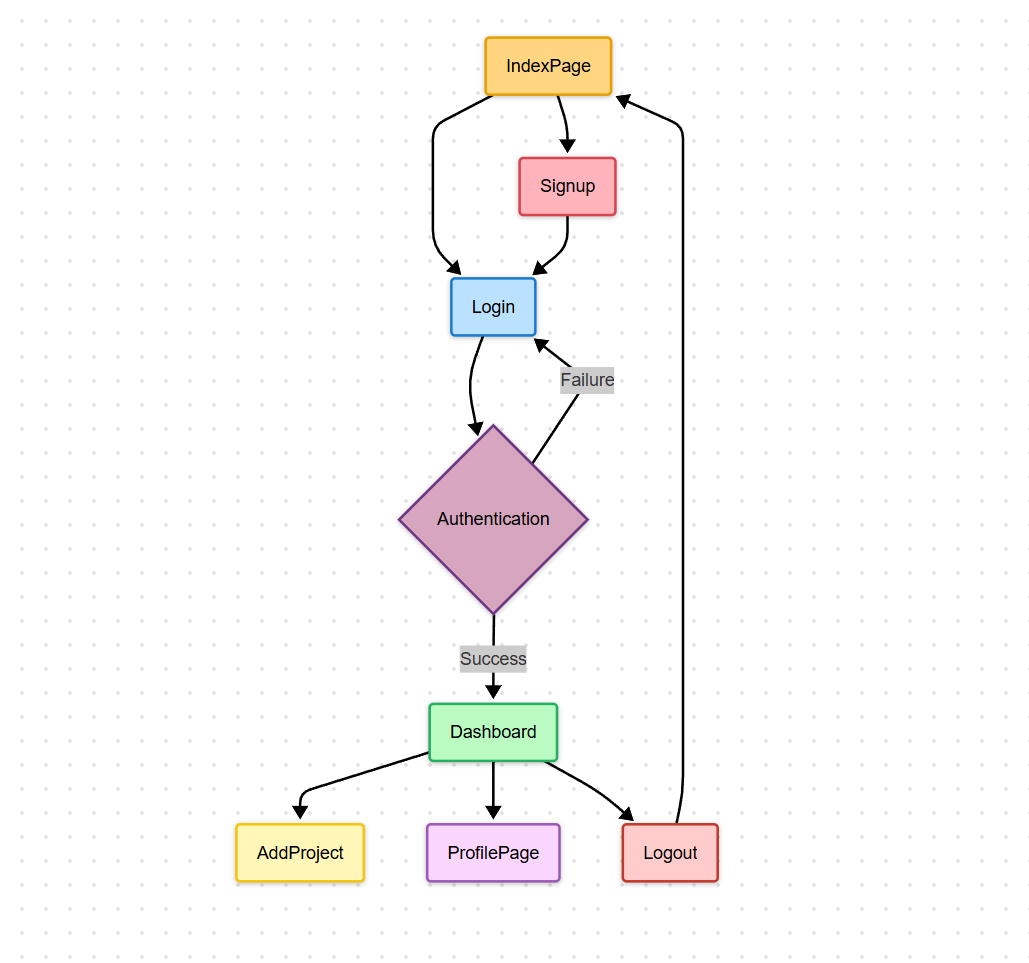
1. View Existing Projects Flow

* Action: User selects "View Projects" from the dashboard.
* Action: The system displays a list of previously submitted projects. The user can filter projects by date, filing status, or title.
* Action: The user reviews project details.
* Decision: If changes or updates are required, the user can edit or reject the project.
  + If the project is approved, the status is updated accordingly.

1. End Point : After completing the desired actions, the user can log out or return to the dashboard.
2. Key Elements

* Start: User login.
* Actions: Creating new projects, reviewing existing projects, submitting, and editing.
* Decisions: Validation of project details, approval/rejection of projects.
* End Points: Confirmation of submission or review, logout.

This flow ensures that the user can easily navigate between creating, reviewing, and submitting IPR projects, with validation at key stages.



## 4.2 Entity-Relationship (ER) Diagram

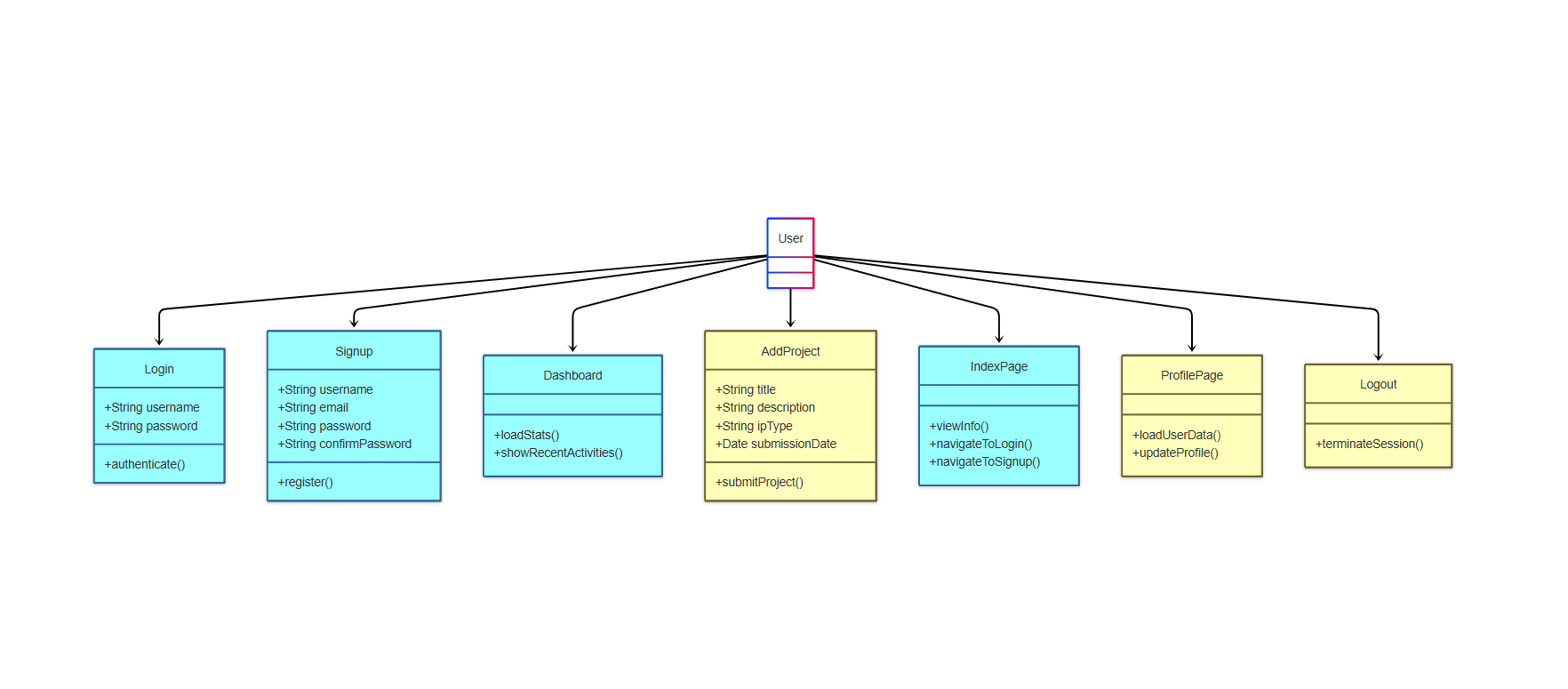
1. **Entities**

* **User**: Represents both normal users and admin users.
  + **Attributes**: User\_ID (PK), Username, Password, User\_Type (Normal/Admin), Email, Date\_Joined
* **IPR Project**: Represents the intellectual property projects submitted by users.
  + **Attributes**: Project\_ID (PK), User\_ID (FK), Project\_Title, Project\_Description, Status (Draft/Submitted/Approved), Filing\_Date
* **Admin**: Represents the system administrators who manage the IPR facilitation center.
  + **Attributes**: Admin\_ID (PK), User\_ID (FK), Role (System/Admin)
* **Project Review**: Represents the process of reviewing IPR projects by the admin.
  + **Attributes**: Review\_ID (PK), Admin\_ID (FK), Project\_ID (FK), Review\_Status (Pending/Approved/Rejected), Review\_Date

1. **Relationships**

* **User-IPR Project**: A **User** can submit many **IPR Projects**.
  + **Cardinality**: One-to-Many (One user can submit many IPR projects)
  + **Foreign Key**: User\_ID in the IPR Project entity references User\_ID in the User entity.
* **Admin-Project Review**: An **Admin** can review multiple **IPR Projects**.
  + **Cardinality**: One-to-Many (One admin can review many IPR projects)
  + **Foreign Key**: Admin\_ID in the Project Review entity references Admin\_ID in the Admin entity.
* **Admin-User Management**: An **Admin** can manage multiple **Users** (add, edit, delete users).
  + **Cardinality**: One-to-Many (One admin can manage many users)
  + **Foreign Key**: Admin\_ID in the User entity references Admin\_ID in the Admin entity.
* **Admin-IPR Project**: An **Admin** can approve or reject **IPR Projects** after reviewing them.
  + **Cardinality**: One-to-Many (One admin can approve/reject many projects)
  + **Foreign Key**: Admin\_ID in the IPR Project entity references Admin\_ID in the Admin entity.

This model captures the key entities involved in managing the intellectual property projects in the system and the relationships between these entities to ensure efficient workflow for the IPR Facilitation Center.



## 4.3 Sequence Diagram

The sequence diagram represents the interaction between a user, system interfaces, and the backend services involved in key processes of the IPR Facilitation Center. It captures the logical flow of actions and data as they occur over time.

1. **Login Sequence**

* **Actors Involved**: User, Login Page, User Authentication Service
* **Narrative**:
  + The user opens the login page and enters their credentials (username and password).
  + The login page sends the credentials to the authentication service.
  + The system validates the credentials against stored data.
  + If valid, the user is granted access and redirected to the dashboard.
  + If invalid, an error message is shown.

1. **Signup Sequence**

* **Actors Involved**: New User, Signup Page, Database
* **Narrative**:
  + A new user navigates to the signup page and fills in the registration form.
  + The signup page sends the user information to the backend.
  + The system checks if the username or email already exists.
  + If unique, the user data is saved in the database and a success message is shown.
  + If not, an error message is displayed.

1. **Project Submission Sequence**

* **Actors Involved**: User, Dashboard, Project Form, Database
* **Narrative**:
  + The user logs in and accesses the dashboard.
  + The user clicks on “Add Project” and fills in the intellectual property project details.
  + The project form sends the data to the backend.
  + The system stores the data in the database.
  + A confirmation of successful submission is returned to the user.

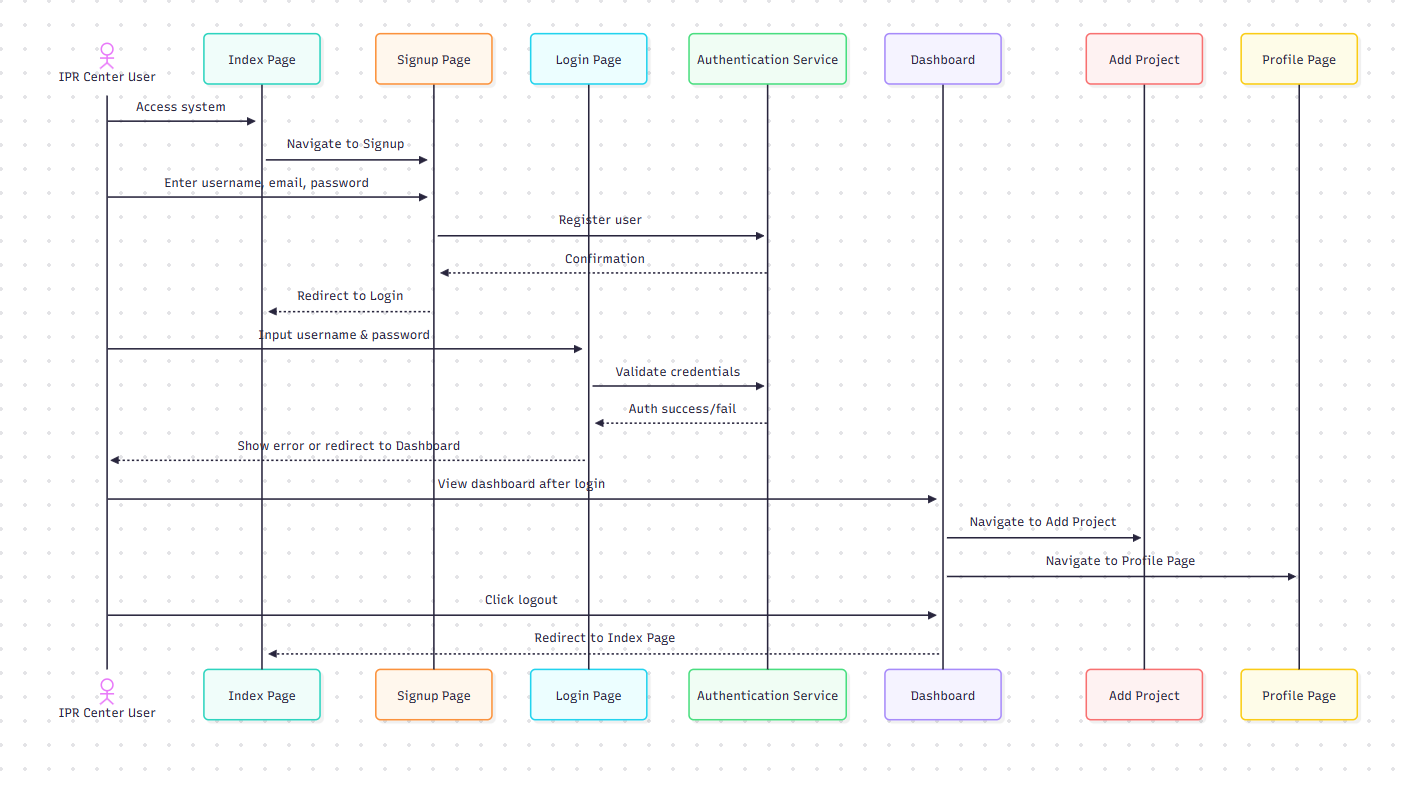
1. **Admin Review Sequence**

* **Actors Involved**: Admin, Dashboard, Project List, Review Interface, Database
* **Narrative**:
  + The admin logs into the system and views a list of submitted projects.
  + The admin selects a project for review and views its details.
  + The admin chooses to approve or reject the project.
  + The status is updated in the database.
  + The user is notified of the outcome.

1. **Logout Sequence**

* **Actors Involved**: User/Admin, System
* **Narrative**:
  + The user or admin clicks the “Logout” button.
  + The system terminates the session and redirects to the login page.

Each of these sequences supports one or more functional requirements outlined in the Software Requirements Specification and ensures a clear and traceable interaction path throughout the system.



# 5 Output Screenshots

