# INTRODUCTION

# 

The Faculty Profile Management System is a comprehensive web application designed to efficiently manage and maintain the profiles of faculty members within an educational institution. This system is built using the powerful combination of Java for the back-end and Angular for the front-end, providing a robust and user-friendly experience.

In educational institutions, managing faculty profiles can become a time-consuming task. It involves handling diverse information about each faculty member, including personal details, educational qualifications, research interests, publications, and more. With the Faculty Profile Management System, these tasks can be streamlined, allowing administrators and faculty themselves to update and access their information easily.

User Authentication and Authorization: The system ensures secure access to data by implementing user authentication and authorization protocols. Faculty members and administrators have their unique login credentials, ensuring data privacy.

Faculty Profile Creation and Management: Faculty members can create and maintain their profiles with relevant information, such as educational qualifications, areas of expertise, work experience, and research publications. They can edit and update their profiles as needed.

Publication Management: Faculty can add their research publications, patents, and projects to their profiles. The system allows them to upload documents or provide external links to published work.

Responsive User Interface: The Angular front-end ensures a responsive and user-friendly interface, accessible across different devices, such as desktops, tablets, and mobile phones.

Admin Dashboard: Administrators have access to a centralized dashboard from which they can manage faculty profiles, review updates, and approve new entries to ensure data accuracy.

# Objectives

The objectives of the Faculty Profile Management System are designed to enhance efficiency, transparency, and collaboration within an educational institution. Here are the key objectives of the system:

Centralized Profile Management: To create a centralized platform where faculty members can easily manage and update their profiles with accurate and up-to-date information, ensuring consistency across the institution.

Streamlined Data Entry: To reduce administrative workload by allowing faculty members to directly input their information, minimizing the need for manual data entry by administrative staff.

Easy Accessibility: To provide a user-friendly interface accessible to faculty members and administrators from various devices, facilitating easy access to faculty profiles anytime and anywhere.

Enhanced Collaboration: To encourage collaboration and networking among faculty members by showcasing their expertise, research interests, and publications, fostering potential interdisciplinary projects and research collaborations.

Improved Decision-Making: To empower administrators with relevant data and insights, enabling them to make informed decisions regarding faculty assignments, research funding, and resource allocation.

# Scope of the project

The scope of the Faculty Profile Management project encompasses the development, implementation, and maintenance of a comprehensive web application to manage and maintain faculty profiles within an educational institution. The project aims to provide a centralized platform that allows faculty members to efficiently manage their profiles while enabling administrators to access relevant data for decision-making. Here are the key aspects of the project scope

The system will allow faculty members to create and maintain their profiles with essential details, including personal information, academic qualifications, work experience, research interests, publications, and any other relevant information.

# Chapter 2

# LITERATURE REVIEW

## Spring Boot Framework

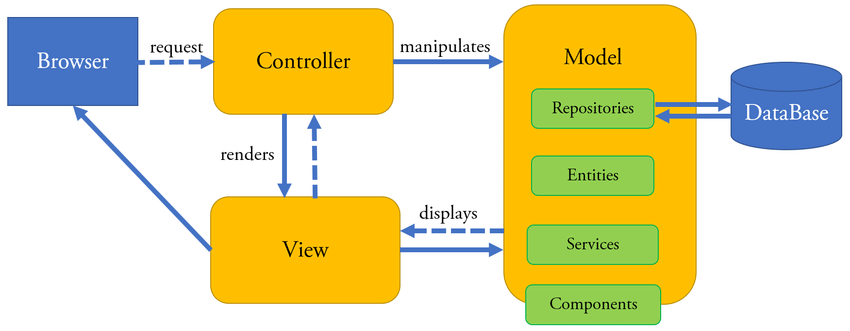
The Spring Boot Framework is a well-known and widely used framework for web and corporate application development. Spring boot uses dependency injection that allows you to create beans via XML, annotations, and Java code. It's a promising platform for creating business and web apps. JPA (Java Persistence API) is a specialised technology that allows you to interface with relational databases. It serves as a bridge between SQL Tables and Java objects. The persistence layer is required for real-world projects that require data storage and retrieval. The author presents how framework integration can speed up system development and significantly cut on coding time. It describes the design and implementation of the Spring and Spring Boot frameworks in the development of Web applications, which reduces the development process and burden of the system, allowing for easier growth and deployment.

* The Model layer wraps the web application data, which is commonly POJO.
* The View layer is in charge of presenting the model data and producing HTML output that the client's browser can understand.
* The third layer, the Controller, is in charge of handling user requests. This layer generates a model, which is then rendered by the view layer.



Spring Data JPA, which is part of the larger Spring Data family, simplifies the creation of JPA-based repositories. This module focuses on improving JPA-based data access layer support. It simplifies the development of Spring-based apps that use data access methods. For a long time, developing a new data access layer for an application has been difficult. To perform simple searches, pagination, and audits, far too much boilerplate code is required. Spring Data JPA aims to significantly simplify data access layer implementation by reducing the amount of effort required to the bare minimum. You write your repository interfaces, including custom locator methods, as a developer, and Spring handles the implementation. Spring Tool Suite IDEA, an integrated development environment (IDE) for creating spring boot microservices, is used throughout the process.

We specify the type of REST API call in the Controller layer by using annotations such as @GetMapping, @PostMapping, @PutMapping, and @DeleteMapping. We've also specified the web service's endpoint. This layer receives the input as a request object and forwards it to the next layer, the Service layer. Request mapping methods available in the Rest Controller include save, find All, findById, and findByFirstName.



To accomplish this, we'll build a package controller that accepts client requests, updates/gets data from a repository, and returns results. Make an H2 database that will be mapped to our entity. We used autowired repository methods that implement the interface CrudRepository in RestController methods annotated with @RequestMapping. The final step is to run the spring boot project and test the results. In this section, we have written logic to set the result data list to our response object, which is in JSON format. When we send a request for a resource with the HTTP method, URI, and request body specified, we receive a response in the form of JSON along with HTTP status codes. The four HTTP methods listed below are commonly used.

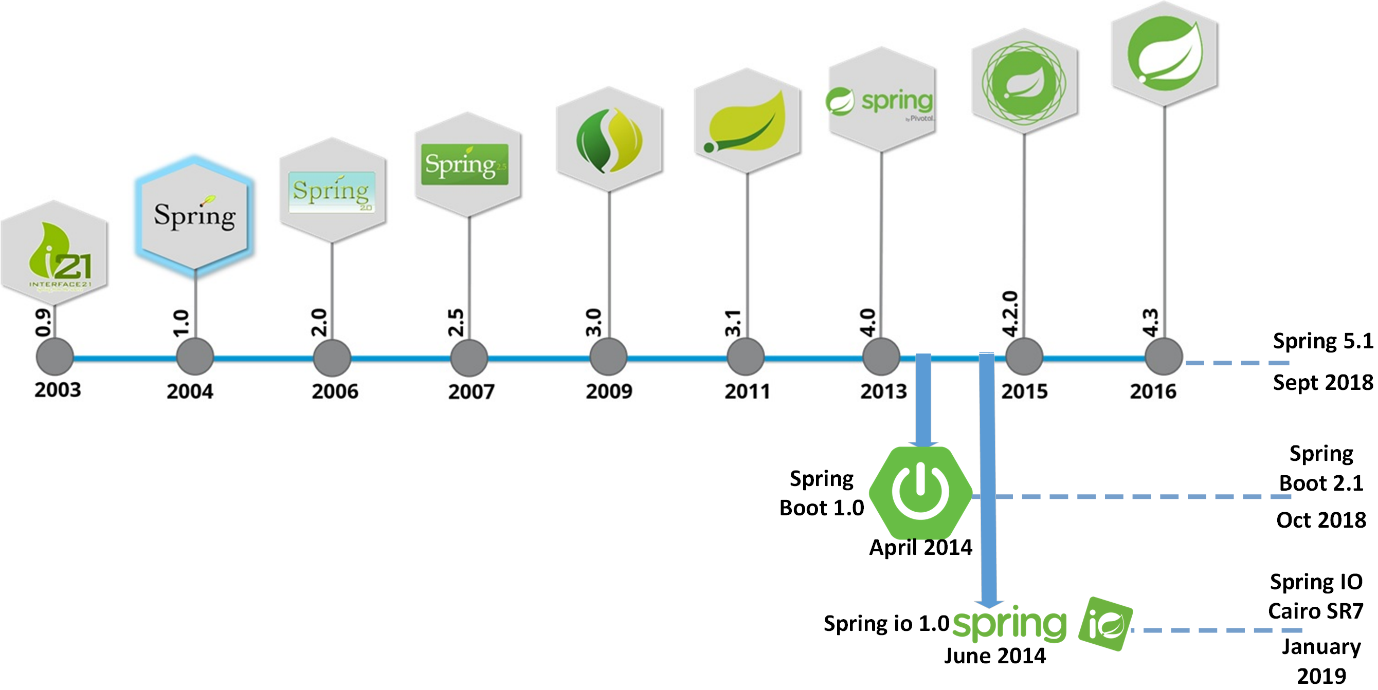
• The GET method is used to access a read-only resource.

• A new resource is created and added using the POST method.

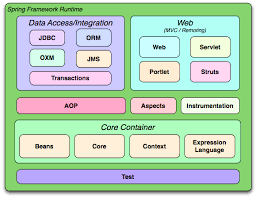
• The DELETE method is used to delete an existing resource.

• The PUT method is used to update an existing resource.

# Spring Boot Versions



# Architecture



# JAVA

Java is a language that allows us to write code once that can be used over and over again. This is very useful because it saves us time and allows us to use other people's code to perform tasks we might otherwise not have the time or knowledge to write for ourselves. Most of the time, we do not even need to see this code or even know how it does its work!

Java is a programming language that has been around a lot longer than Android. It is an object-oriented language.

The primary motivation of this language was the need for a platform-independent (i.e. architecture neutral) language that could be used to create software to be embedded in various consumer electronic devices.

• Java is a programmer’s language

• Java is cohesive and consistent

• Except for those constraint imposed by the Internet environment.

• Java is a known language, developers know it and don't have to learn it

• It is harder to shoot yourself with Java than with C/C++ code since it has no pointer arithmetic

• It runs in a VM, so no need to recompile it for every phone out there and easy to secure

• There are a large number of development tools for Java

• Several mobile phones already used Java ME, so Java was known in the industry

• There are a large number of developers already proficient in Java.

• Java has huge open source support, with many libraries and tools available to make developers life easier.

• Java protects you from many of the problems inherent in native code, like memory leaks, bad pointer usage, etc.

• Java allows them to create sandbox applications, and create a better security model so that one bad App can't take down your entire OS.

## 2.2.1 Java Architecture

Java architecture provides a portable, robust, high performing environment for development. Java provides portability by compiling the byte codes for the Java Virtual Machine, which is then interpreted on each platform by the run-time environment. Java is a dynamic system, able to load code when needed from a machine in the same room or across the planet.

Compilation of code - When you compile the code, the Java compiler creates machine code (called byte code) for a hypothetical machine called Java Virtual Machine (JVM). The JVM is created for overcoming the issue of probability. The code is written and compiled for one machine and interpreted on all machines. This machine is called Java Virtual Machine.

# HTML

HTML (Hypertext Markup Language) is the standard language used to create and structure the content of web pages. It consists of a series of elements and tags that define the structure and layout of a web document. Below, I'll provide you with a basic HTML document template that you can use as a starting point for creating your web pages:

<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Your Page Title</title>

</head>

<body>

<header>

<h1>Welcome to Your Website</h1>

</header>

<nav>

<ul>

<li><a href="#home">Home</a></li>

<li><a href="#about">About</a></li>

<li><a href="#services">Services</a></li>

<li><a href="#contact">Contact</a></li>

</ul>

</nav>

<main>

<section id="home">

<h2>Home Section</h2>

<p>Welcome to our website! This is the home section</p>

</section>

<section id="about">

<h2>About Section</h2>

<p>Learn more about our company and what we do</p>

</section>

<section id="services">

<h2>Services Section</h2>

<p>Explore the services we offer to our clients</p>

</section>

<section id="contact">

<h2>Contact Section</h2>

<p>Get in touch with us using the contact form</p>

</section>

</main>

<footer>

<p>&copy; 2023 Your Company. All rights reserved</p>

</footer>

</body>

</html>

In this HTML template, we have a basic structure with essential HTML elements:

The <!DOCTYPE html> declaration specifies that this is an HTML5 document.

The <html> element is the root element of the HTML document.

The <head> element contains metadata about the document, such as the character encoding, viewport settings, and the page title.

The <body> element holds the visible content of the web page.

Inside the <body>, we have a header, navigation, main content, and footer sections.

The <header> element typically contains the site's logo, title, or introductory text.

The <nav> element represents the site's navigation menu.

The <main> element wraps the main content of the page.

<section> elements define different sections of the web page.

The <footer> element contains copyright information or any other footer content.

# JavaScript

JavaScript is a high-level, dynamic, and versatile programming language used to add interactivity, functionality, and dynamic content to web pages. Below, I'll provide you with a basic JavaScript template that you can use as a starting point for writing JavaScript code:

<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Your Page Title</title>

</head>

<body>

<h1>Hello, JavaScript</h1

<script>

const name = "John";

const greeting = "Hello, " + name + "!";

alert(greeting);

</script>

</body>

</html>

In this template, we have a basic HTML structure that includes a simple JavaScript code snippet inside the <script> tag:

The <script> tag is used to embed JavaScript code directly into the HTML document.

JavaScript code can be placed between the opening <script> and closing </script> tags.

In this example, we have a basic JavaScript code that creates a variable name with the value "John" and another variable greeting that concatenates the name with a greeting message.

The alert() function displays a pop-up dialog box with the concatenated greeting when the page loads.

# CSS

CSS (Cascading Style Sheets) is a style sheet language used to control the presentation and layout of HTML documents. It allows you to define the visual appearance of elements on a web page. Below, I'll provide you with a basic CSS template that you can use as a starting point for writing CSS code:

h1 {

color: blue;

font-size: 32px;

text-align: center;

}

p {

color: green;

font-size: 16px;

text-align: justify;

}

In this CSS file, we have defined two CSS rules:

The h1 rule applies styles to all <h1> elements on the page. It sets the text colour to blue, font size to 32 pixels, and centres the text horizontally using text-align.

The p rule applies styles to all <p> elements on the page. It sets the text colour to green, font size to 16 pixels, and justifies the text (aligns both left and right edges).

When you open the HTML document in a web browser, it will apply the CSS styles from the "styles.css" file, and you'll see the changes in the appearance of the <h1> and <p> elements.

CSS provides an extensive range of properties and selectors that allow you to control the layout, colours, fonts, and other visual aspects of your web page. You can experiment with various CSS rules to achieve the desired visual presentation for your HTML elements.

# Chapter 3

# SYSTEM REQUIREMENTS

## EXISTING SYSTEM

The existing system in faculty profile management might vary from one educational institution to another, depending on their size, resources, and technological capabilities. However, in many cases, traditional faculty profile management systems involve manual processes and may include the following elements:

1. Physical Files and Documents: In some institutions, faculty profiles are managed using physical files and documents. Each faculty member has a file containing their educational qualifications, work experience, research publications, and other relevant information. This approach can be time-consuming and prone to errors.

2. Spreadsheets or Word Documents: Some institutions use spreadsheets or word documents to maintain faculty profiles. Faculty members are required to fill out forms with their details, which are then manually updated by administrative staff. This method can lead to data inconsistency and challenges in retrieving information.

3. Departmental Databases: Larger institutions may have departmental databases to store faculty information. These databases might be designed in-house or use generic database management systems. However, these systems may lack integration with other institutional databases and applications.

4. Institutional Websites: Many educational institutions maintain faculty profiles on their official websites. Faculty members' information is manually updated on webpages, often with limited interactivity and search capabilities.

Challenges in the Existing System:

* Manual data entry and management can lead to data inconsistency and errors.
* Limited accessibility and interactivity for faculty members.
* Difficulty in searching and filtering faculty profiles efficiently.
* Inefficiency in handling updates and changes in a timely manner.
* Lack of integration with other institutional systems.
* Limited data analytics capabilities for decision-making.

Given these challenges, many educational institutions are exploring modern faculty profile management systems that leverage technology and automation to streamline the process, enhance data accuracy, and provide a better user experience for both faculty members and administrators. Such systems often utilize web-based applications, databases, and integration with other institutional systems for a more efficient and comprehensive faculty profile management solution.

# PROPOSED SYSTEM

The proposed system for faculty profile management aims to address the limitations of the existing system and provide a more efficient, user-friendly, and comprehensive solution. The proposed system leverages modern technologies and best practices to streamline faculty profile management, enhance collaboration, and improve decision-making. Here are the key features of the proposed system:

Web-Based Application: The proposed system will be a web-based application accessible through standard web browsers. This ensures easy access and usability across different devices, such as desktops, laptops, tablets, and smartphones.

User-Friendly Interface: The system will have an intuitive and user-friendly interface, making it easy for both faculty members and administrators to navigate and update profiles efficiently.

Self-Service Profile Management: Faculty members will have the ability to create, update, and maintain their profiles directly. They can add educational qualifications, research interests, publications, and other relevant information on their own.

# FUNCTIONAL REQUIREMENTS

It defines the functionality of the software and the constraints on it. Functional requirements describe the features, functioning, and usage of a product/system/software from the perspective of the product and its user. The document will not only define the product functions, user characteristics and constraints but also serve as a basis for software design document

## MODULES OF THE PROJECT:

1. Login
2. Registration
3. Admin Dashboard
4. User Dashboard
5. Add Faculty -> faculty details include

* first name
* last name
* email id
* phone number
* experience
* qualification
* research publication
* research guidance
* achievements

1. List Faculty
2. Delete
3. Update
4. Saving faculty details and preference in MYSQL data base
5. Menu items

* About
* Contact
* Login
* Register
* Faculty list
* Add faculty

# NON-FUNCTIONAL REQUIREMENTS

Non-functional requirements in faculty profile management refer to the qualities and characteristics that define how the system should perform and behave rather than specifying its functionality. These requirements address aspects such as performance, security, usability, reliability, and scalability. Here are some non-functional requirements relevant to faculty profile management:

1. Performance:

* The system should load faculty profiles quickly, ensuring minimal waiting times for users.
* Search and filtering operations should be efficient, delivering results promptly.
* Response times for profile updates and data retrieval should be within acceptable limits.

2. Scalability:

* The system should be able to handle an increasing number of faculty profiles and users as the institution grows.
* It should support concurrent access by multiple users without significant degradation in performance.

3. Security:

* Faculty profile data should be stored securely, protecting sensitive information from unauthorized access or breaches.
* User authentication and authorization mechanisms should be in place to ensure that only authorized users can access and modify profiles.

4. Usability:

* The user interface should be intuitive and easy to navigate for both faculty members and administrators.
* Profile update forms and data entry fields should be clear and straightforward to use.
* Error messages and feedback should be informative and user-friendly.

5. Reliability:

* The system should be highly reliable, ensuring minimal downtime and providing consistent access to faculty profiles.
* Data integrity should be maintained, preventing loss or corruption of faculty profile data.

6. Compatibility:

* The system should be compatible with various web browsers and devices to ensure a consistent user experience.
* It should support common operating systems and network environments.

7. Accessibility:

* The system should adhere to accessibility guidelines (e.g., WCAG) to cater to users with disabilities.
* Faculty members with diverse abilities should be able to access and use the system effectively.

# FEASIBILITY STUDY

A feasibility study for faculty profile management aims to assess the practicality, viability, and potential success of implementing a new system or upgrading an existing one. It involves analysing various aspects, including technical, economic, operational, legal, and scheduling factors. Let's look at the key areas to consider in the feasibility study for faculty profile management:

1. Technical Feasibility:

* Assess the institution's existing technical infrastructure and determine if it can support the new system's requirements.
* Evaluate the availability of skilled technical resources required for development, implementation, and maintenance.
* Identify any potential technical challenges or limitations that may arise during the system's implementation.

2. Economic Feasibility:

* Conduct a cost-benefit analysis to determine if the benefits of the new system outweigh the costs.
* Evaluate the initial investment required for system development, hardware, software, and training.
* Consider the potential cost savings from increased efficiency, reduced paperwork, and improved decision-making.

3. Operational Feasibility:

* Evaluate the impact of the new system on the day-to-day operations of the faculty and administrative staff.
* Consider how the system will integrate with existing processes and workflows.
* Assess the level of acceptance and support from faculty members and administrators.

4. Legal and Regulatory Feasibility:

* Ensure that the system complies with relevant legal and regulatory requirements, such as data protection and privacy laws.
* Identify any potential legal or compliance risks associated with the system.

5. Scheduling Feasibility:

* Develop a realistic timeline for system development, testing, and implementation.
* Consider any external dependencies or constraints that could affect the project schedule.

6. Security and Privacy Feasibility:

* Evaluate the security measures implemented in the system to safeguard faculty profile data.
* Ensure that the system adheres to institutional data privacy policies and relevant regulations.

7. User Acceptance and Training:

* Assess the willingness of faculty members and administrators to adopt the new system.
* Plan for comprehensive training and support to ensure users can effectively use the system.

8. Scalability:

* Evaluate whether the proposed system can handle future growth in the number of faculty members and profiles.
* Consider how the system can accommodate new features or requirements in the future.

9. Risk Analysis:

* Identify potential risks associated with the implementation of the new system and develop mitigation strategies.

Based on the findings of the feasibility study, stakeholders can make informed decisions about whether to proceed with the faculty profile management system project, make adjustments to the plan, or explore alternative solutions. The feasibility study provides valuable insights that contribute to the project's success and help ensure that the chosen solution aligns with the institution's goals and resources.

# SOFTWARE REQUIREMENTS

|  |  |
| --- | --- |
| **Operating System** | Windows 7 or higher |
| **Languages Used** | Java, HTML, Java Script, CSS |
| **Frameworks** | Spring Boot, Angular |
| **Development Environment (IDE)** | STS, Visual Code |
| **Database** | MYSQL |

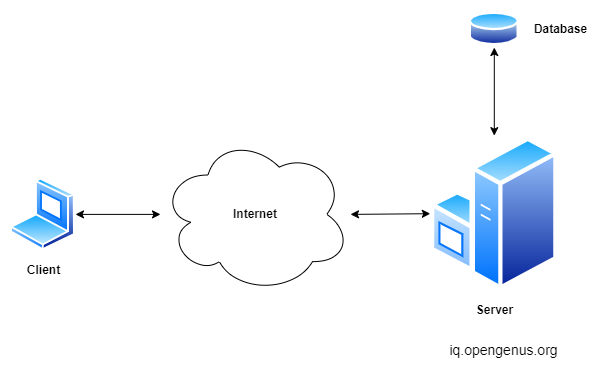
# HARDWARE REQUIREMENTS

|  |  |
| --- | --- |
| **Hard disk** | Minimum of 40GB |
| **RAM** | 4 GB or higher |

# Chapter4

# SYSTEM ARCHITECTURE

The system architecture for faculty profile management is a high-level representation of how the various components and modules of the system interact and collaborate to achieve the desired functionality. The architecture should be designed to ensure scalability, maintainability, and performance. Here's a proposed system architecture for faculty profile management:



# Flow

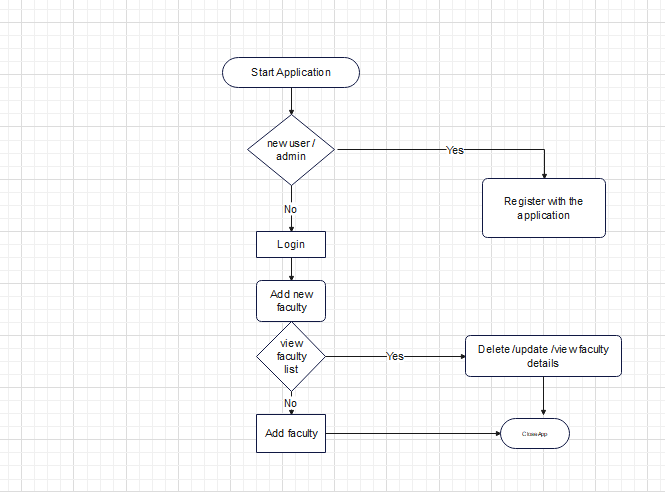
Flowcharts are graphical representations of a process or system, using symbols and arrows to illustrate the flow of steps and decisions. For a faculty profile management system, you can use the following flowchart symbols and notations:

## Flowchart Symbols and notation:

These flowchart shapes and symbols are some of the most common types you'll find in most flowchart diagrams.

| **Flowchart Symbol** | **Name** | **Description** |
| --- | --- | --- |
| Process Flowchart Symbol | Process symbol | Also known as an “Action Symbol,” this shape represents a process, action, or function. It’s the most widely-used symbol in flowcharting. |
| Start/End Flowchart Symbol | Start/End symbol | Also known as the “Terminator Symbol,” this symbol represents the start points, end points, and potential outcomes of a path. Often contains “Start” or “End” within the shape. |
| Document Flowchart Symbol | Document symbol | Represents the input or output of a document, specifically. Examples of and input are receiving a report, email, or order. Examples of an output using a document symbol include generating a presentation, memo, or letter. |
| Decision Flowchart Symbol | Decision symbol | Indicates a question to be answered — usually yes/no or true/false. The flowchart path may then split off into different branches depending on the answer or consequences thereafter. |
| Connector Flowchart Symbol | Connector symbol | Usually used within more complex charts, this symbol connects separate elements across one page. |
| Off-page Connector Flowchart Symbol | Off-Page Connector/Link symbol | Frequently used within complex charts, this symbol connects separate elements across multiple pages with the page number usually placed on or within the shape for easy reference. |
| Input/Output Flowchart Symbol | Input/Output symbol | Also referred to as the “Data Symbol,” this shape represents data that is available for input or output as well as representing resources used or generated. While the paper tape symbol also represents input/output, it is outdated and no longer in common use for flowchart diagramming. |

If the user is new to the app, then he has to register with the app. The app requires details like name, mail id, contact number, password for registration. Upon successful registration, the user can add faculty and, delete faculty, update faculty and view all faculty list.



# Chapter 5

# SYSTEM DESIGN

The design document that we develop during this phase is the blueprint of the software. It describes how the solution to the customer problem is to be built. The design activity begins when the requirements document for the software to be developed is available. While the requirements specification activity is entirely in the problem domain, design is the first step in moving from the problem domain toward the solution domain. Design is essentially the bridge between requirements specification and the final solution for satisfying the requirements.

The design of a system is essentially a blueprint or a plan for a solution for the system. The design process for software systems, often, has two levels. At the first level, the focus is on deciding which modules are needed for the system, the specifications of these modules, and how the modules should be interconnected. This is what is called the system design or top-level design. In the second level, the internal design of the modules, or how the specifications of the module can be satisfied, is decided. This design level is often called detailed design or logic design. Detailed design essentially expands the system design to contain a more detailed description of the processing logic and data structures so that the design is sufficiently complete for coding

# DATA FLOW DIAGRAM

A data flow diagram (DFD) maps out the flow of information for any process or system. It uses defined symbols like rectangles, circles and arrows, plus short text labels, to show data inputs, outputs, storage points and the routes between each destination. Data flowcharts can range from simple, even hand-drawn process overviews, to in-depth, multi-level DFDs that dig progressively deeper into how the data is handled. They can be used to analyse an existing system or model a new one. Like all the best diagrams and charts, a DFD can often visually “say” things that would be hard to explain in words, and they work for both technical and nontechnical audiences, from developer to CEO.

# DFD rules and tips

· Each process should have at least one input and an output.

· Each data store should have at least one data flow in and one data flow out.

· Data stored in a system must go through a process.

· All processes in a DFD go to another process or a data store.

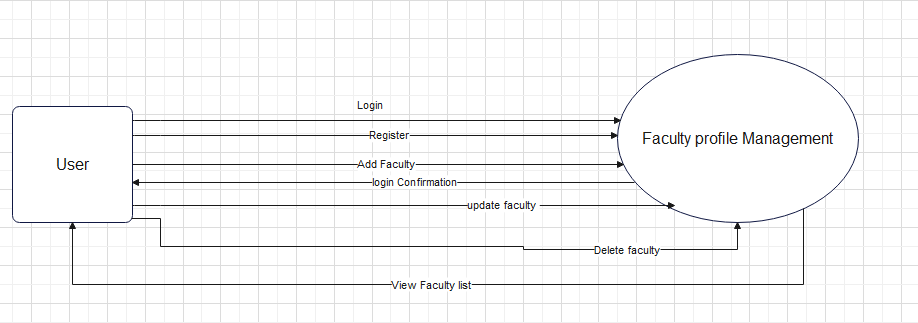
# DFD levels

A data flow diagram can dive into progressively more detail by using levels and layers, zeroing in on a particular piece. DFD levels are numbered 0, 1 or 2, and occasionally go to even Level 3 or beyond. The necessary level of detail depends on the scope of what you are trying to accomplish.

DFD Level 0 is also called a Context Diagram. It’s a basic overview of the whole system or process being analysed or modelled. It’s designed to be an at-a-glance view, showing the system as a single high-level process, with its relationship to external entities. It should be easily understood by a wide audience, including stakeholders, business analysts, data analysts and developers.

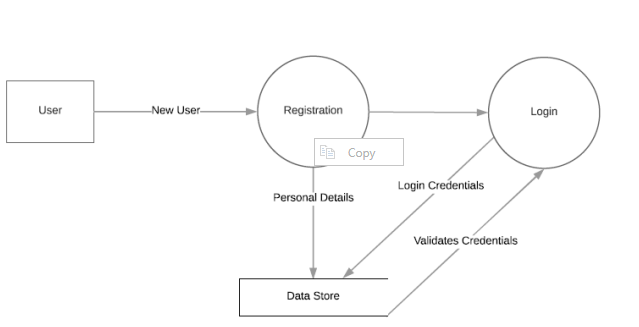
DFD Level 1 provides a more detailed breakout of pieces of the Context Level Diagram. You will highlight the main functions carried out by the system, as you break down the high-level process of the Context Diagram into its sub processes.

DFD Level 2 then goes one step deeper into parts of Level 1. It may require more text to reach the necessary level of detail about the system’s functioning.

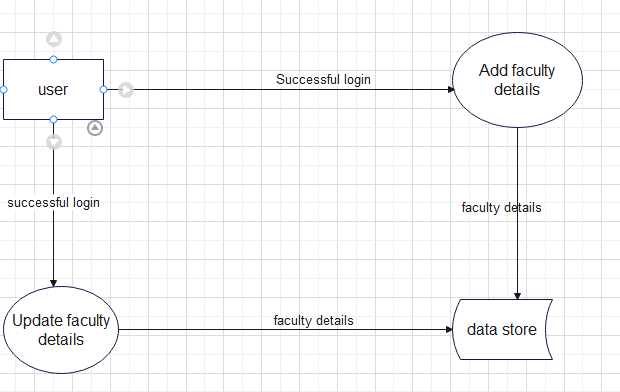


Level 0 DFD

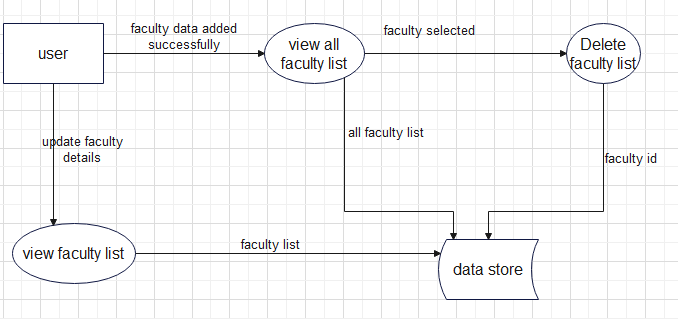
The Level 0 DFD shows the Expense Tracker App as a high-level process with the main entity User.



Level 1 DFD – Login and Registration



Level 1 DFD – Adding Faculty Details

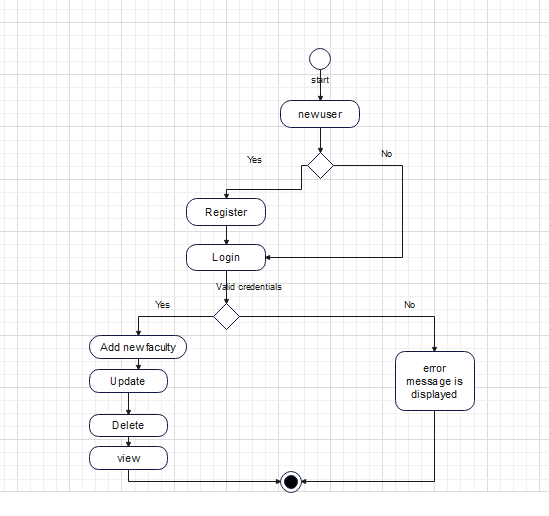


Level 1 DFD – Viewing Faculty list

In the Level 1 DFD the entire system is divided into various processes like Login, Registration, Adding Faculty, Update faculty details, Viewing faculty list.

# ACTIVITY DIAGRAMS

The major activities of the Expense Tracker application are depicted here.



# USE CASE DIAGRAMS

A use case diagram is a graphical representation of the user’s interaction with the system. It can portray the different types of users of a system and the various ways they interact with the system. Use cases are diagrammed to be easily understood, no matter who is looking at the diagram.

A use case diagram can summarize the details of your system's users (also known as actors) and their interactions with the system. An effective use case diagram helps to represent:

1. Scenarios in which your system or application interacts with people, organizations, or external systems

2. Goals that it helps those entities (known as actors) achieve

3. The scope of your system

Purpose of Use Case Diagrams:

1. To demonstrate the different ways that a user might interact with a system.

2. To gather the requirements of a system.

3. To get an outside view of a system.

4. To identify the external and internal factors influencing the system.

Use Case Diagram Components:

1. Actors - the users that interact with a system. An actor can be a person, an organization, or an outside system that interacts with your application or system. They must be external objects that produce or consume data.

2. System - a specific sequence of actions and interactions between actors and the system. A system may also be referred to as a scenario.

3. Goals - the end result of most use cases. A successful diagram should describe the activities and variants used to reach the goal.

Use Case Diagram Symbols and Notation:

Use case diagrams consists of actors, use cases and their relationships.

1. Use cases - Use cases are the horizontally shaped ovals. This represents the different uses that a user might need.

2. Actors - represented by stick figure people and are the people actually employing the use cases

3. Associations - represented by a line between actors and use cases.

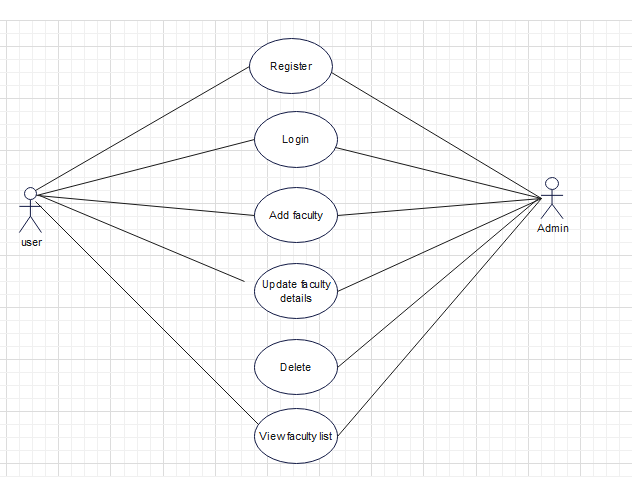
Advantages of Use Case Diagrams:

1. A use case diagram helps provide a high-level view of the system

2. It provides a simplified graphical representation of what the system must actually do

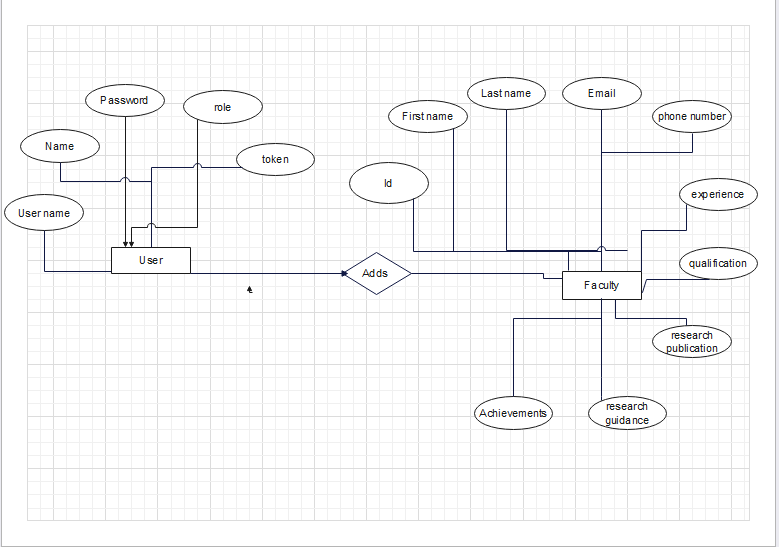
3. Use case diagrams are a good communication tool for stakeholders

User has to register with the app with his details and then he can Login to the system. After logging in a user can add his monthly income and expenses. In addition to this he can also view expenses and reports. The app also provides option to mail and SMS monthly expenses.



# ER DIAGRAM

An Entity Relationship (ER) Diagram is a type of flowchart that illustrates how “entities” such as people, objects or concepts relate to each other within a system. ER Diagrams are most often used to design or debug relational databases in the fields of software engineering, business information systems, education and research. Also known as ERDs or ER Models, they use a defined set of symbols such as rectangles, diamonds, ovals and connecting lines to depict the interconnectedness of entities, relationships and their attributes. They mirror grammatical structure, with entities as nouns and relationships as verbs.



# Chapter 6

# IMPLEMENTATION

# 6.1 SOFTWARE TOOLS USED

* Spring tool suite
* Visual code

# 6.2 PROGRAMMING TECHNOLOGY

* Java
* HTML
* CSS
* JavaScript

# 6.3 SPRING BOOT AND ANGULAR PROJECT IMPLEMENTATION REQUIREMENTS

## **6.3.1The Spring Boot Application**

A Spring Boot application is a Java-based web application built using the Spring Boot framework. Spring Boot simplifies the process of creating production-ready applications by providing an opinionated setup and convention-over-configuration approach. It includes embedded servers, auto-configuration, and dependency management to facilitate rapid development.

Here are the key components and features of a typical Spring Boot application:

Main Application Class:

Every Spring Boot application starts with a main application class annotated with @SpringBootApplication.

This class serves as the entry point for the application and enables Spring Boot auto-configuration.

Controllers:

Controllers are classes annotated with @RestController or @Controller that handle HTTP requests and define API endpoints.

They receive incoming requests, process data, invoke services, and return responses.

Services:

Services are classes annotated with @Service that contain business logic and handle data processing.

They are responsible for performing operations requested by the controllers.

Data Access Layer:

Spring Boot applications often use Spring Data JPA to interact with databases.

Repositories, annotated with @Repository, define CRUD (Create, Read, Update, Delete) operations for database entities.

Configuration:

Spring Boot applications use annotations like @Configuration, @Bean, and @Autowired for configuration and dependency injection.

Custom configurations can be provided to override default behavior.

Error Handling:

Spring Boot provides mechanisms for handling errors and exceptions, such as @ExceptionHandler and @ControllerAdvice.

Security:

Spring Security is commonly used for implementing security features like authentication and authorization.

Application Properties:

Configuration settings for the application are typically provided in the application. Properties or application.yml file.

It allows for easy externalized configuration without changing the code.

Build and Deployment:

Spring Boot applications are typically packaged as JAR files, making them easy to deploy and run on various platforms.

Tools like Maven or Gradle are commonly used for building and managing dependencies.

## **6.3.2 Angular Application**

Angular is a popular open-source web application framework maintained by Google. It is written in TypeScript and allows developers to build dynamic, single-page web applications (SPAs) with ease. Angular provides a robust set of tools and features for building scalable and maintainable applications.

1. Component-Based Architecture:

Angular follows a component-based architecture, where the application is divided into reusable and modular components.

Each component encapsulates its own template, styles, and behaviour, making it easy to manage and maintain.

2. Two-Way Data Binding:

Angular offers two-way data binding, enabling automatic synchronization of data between the model and the view.

When the data changes in the model, the view is automatically updated, and vice versa.

3. Directives:

Directives are instructions in the DOM that modify its behaviour or appearance.

Angular has built-in directives like `ngIf`, `ngFor`, and `ngStyle`, and allows developers to create custom directives as well.

4. Dependency Injection (DI):

Angular dependency injection system helps manage the dependencies between different components and services.

It promotes loose coupling and facilitates unit testing.

5. Routing:

Angular provides a powerful routing mechanism that allows developers to define navigation paths and load different components based on URLs.

This is essential for building SPAs with multiple views.

6. Forms and Validation:

Angular offers comprehensive support for handling forms and implementing client-side form validation.

It provides features like form controls, validators, and form groups.

7. HTTP Client:

Agular’s HTTP client module makes it easy to perform HTTP requests and handle responses, enabling communication with APIs and servers.

8. Observables:

Angular uses RxJS observables to manage asynchronous operations, such as handling HTTP responses or managing event streams.

9. CLI (Command-Line Interface):

- Angular CLI provides a command-line interface that automates the setup and scaffolding of Angular projects and components.

Angular is widely used for building various types of applications, including enterprise applications, dashboards, e-commerce platforms, and more. Its rich set of features, strong community support, and regular updates make it a popular choice for modern web development.

## 6.3.3Application SQL Databases

MySQL is an open-source relational database management system (RDBMS) that is widely used for storing, managing, and retrieving data. It is a popular choice for web applications, including those built with Spring Boot and Angular, due to its ease of use, performance, and scalability.

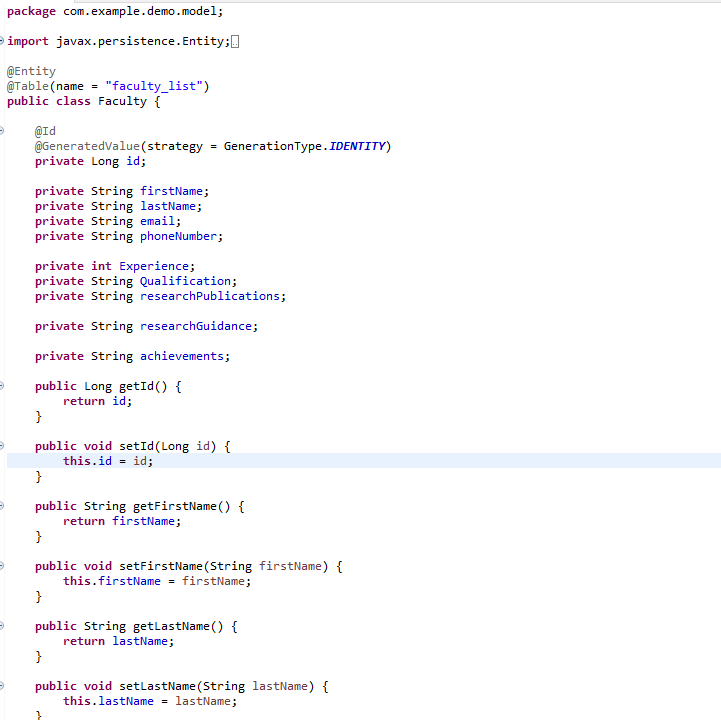
To use MySQL with your Spring Boot and Angular application, you can configure the Spring Boot application to connect to the MySQL database using Spring Data JPA or JDBC. The Angular application can then communicate with the Spring Boot backend to perform database operations through RESTful APIs.

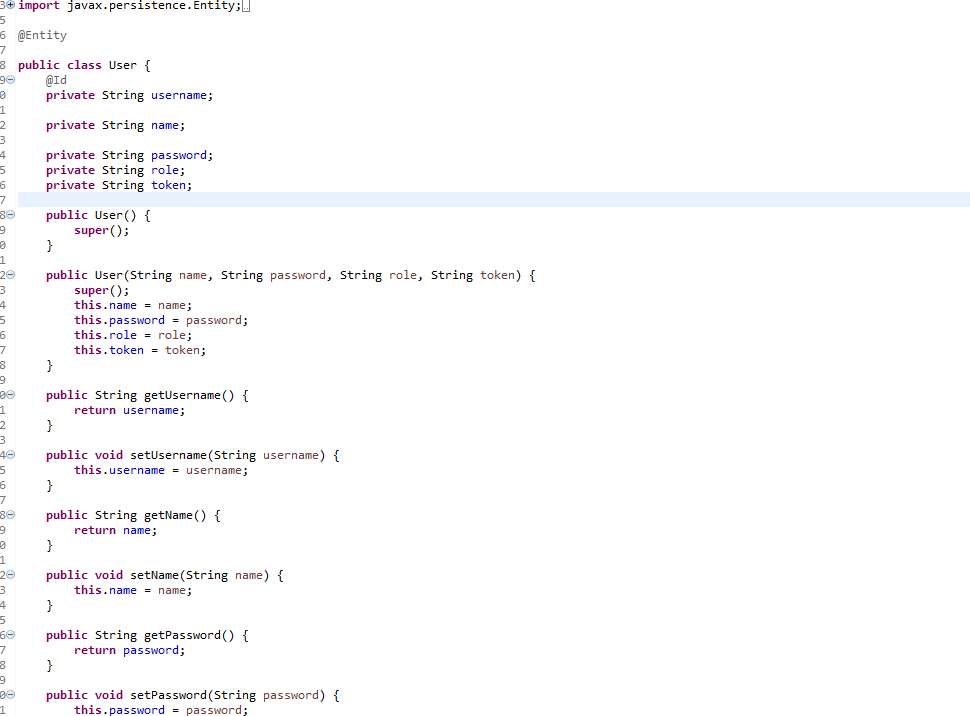
# 6.4 CODE SNIPPETS

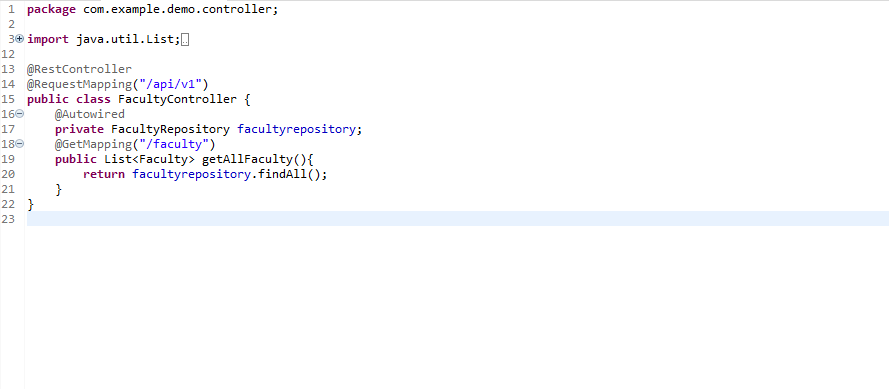
Login And Register













# Chapter 7

# SOFTWARE TESTING

## 7.1 INTRODUCTION

Software testing is a process of executing a program or application with the intent of finding the software bugs.

Software testing is a critical element of software quality assurance and represents the ultimate process to ensure the correctness of the product. The quality product always enhances the customer confidence in using the product thereby increasing the business economics. In other words, a good quality product means zero defects, which is derived from a better-quality process in testing.

Testing the product means adding value to it by raising the quality or reliability of the product. Raising the reliability of the product means finding and removing errors. Hence one should not test a product to show that it works; rather, one should start with the assumption that the program contains errors and then test the program to find as many of the errors as possible.

The main objective of testing is to find defects in requirements, design, documentation, and code as early as possible. The test process should be such that the software product that will be delivered to the customer is defect less. All Tests should be traceable to customer requirements. Test cases must be written for invalid and unexpected, as well as for valid and expected input conditions.

A necessary part of a test case is a definition of the expected output or result. A good test case is one that has high probability of detecting an as-yet undiscovered error.

## Manual Testing

Manual testing includes testing a software manually, i.e., without using any automated tool or any script. In this type, the tester takes over the role of an end-user and tests the software to identify any unexpected behaviour or bug. There are different stages for manual testing such as unit testing, integration testing, system testing, and user acceptance testing.

Testers use test plans, test cases, or test scenarios to test a software to ensure the completeness of testing. Manual testing also includes exploratory testing, as testers explore the software to identify errors in it.

Different stages for manual testing:

## Unit Testing

This type of testing is performed by developers before the setup is handed over to the testing team to formally execute the test cases. Unit testing is performed by the respective developers on the individual units of source code assigned areas. The developers use test data that is different from the test data of the quality assurance team.

Tests that are performed during the unit testing in GMS app are explained below:

1) Module Interface test: In module interface test, it is checked whether the information is properly flowing in to the program unit (or module) and properly happen out of it or not.

The user registration details should be available from the layout to the corresponding controller and from the controller it should flow to the model.

2) Boundary conditions: It is observed that much software often fails at boundary related conditions. That’s why boundary related conditions are always tested to make safe that the program is properly working at its boundary condition’s.

In case of if...else if... else... construct all the conditions are checked in the app. In case of loops, it is checked to see that the loops are not infinite and terminate once the condition becomes false.

3) Error handling paths: These are tested to review if errors are handled properly.

1. Validation during login (Checking for wrong credentials)

2. Validation of password during registration (Password should adhere to password policy - minimum 8 characters with a number and special character)

3. Validation for Category, Amount, Description when adding expense

Integration Testing

Integration testing is defined as the testing of combined parts of an application to determine if they function correctly. Integration testing can be done in two ways: Bottom-up integration testing and Top-down integration testing. In this project we have followed the Bottom-up integration method.

Here testing begins with unit testing, followed by tests of progressively higher-level combinations of units called modules or builds.

Once all the different modules were integrated in the app, the app was tested for the following:

1. Transition from one screen to another

2. Data from the layouts is getting saved properly in the database.

3. Data is retrieved properly from the database and displayed in the layouts.

4. SMS and mail are sent properly when user selects this option from menu.

5. Menu items are working properly.

6. Reports are generated with accurate values

System Testing

System testing tests the system as a whole. Once all the components are integrated, the application as a whole is tested rigorously to see that it meets the specified Quality Standards.

The app was installed on an Android mobile and all the features were tested rigorously for all possible inputs. Different test cases were executed to see if the app behaved as executed and there were no crashes and unexpected behaviour.

# 7.4 TEST CASES

## 7.4.1 Functional Test Cases

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Test  Case  ID | Test Case  Name | Test case Description | step | I/P given | Expected O/P | Actual O/P | Test Status P/F |
| TC01 | Login | To verify that the user has entered valid username and password | Login with  valid username and password | valid username and password | Login Successful | Login Successful | Pass |
|  | Login | To verify that the user has entered valid username and password | Login with Invalid username and password | Invalid username and password | Login Successful | Appropriate  Error message is displayed | Pass |
| TC02 | Registration | To verify that the user has registered by entering valid details | Enter all the valid user details | Valid  details | Registered successfully | Registered successfully | Pass |
|  | Registration | To verify that the user has registered by entering valid details | If entered details are not valid or some of the details are missing | Invalid details | Registration unsuccessful | Appropriate  Error  Message is displayed | Pass |
| TC03 | Add Faculty | To verify that the  user has entered, name,  email,  phone number,  experience,  qualification | All the required details are entered | Valid details | Faculty is added successfully | Faculty is added successfully | Pass |
| TC04 | Update  Faculty | To verify that the  user has entered, name,  email,  phone number,  experience,  qualification | All the required details are entered | Valid details | Faculty is Updated successfully | Faculty is Updated successfully | Pass |

# Chapter 8

# ANNEXURE/SCREEN SHOTS

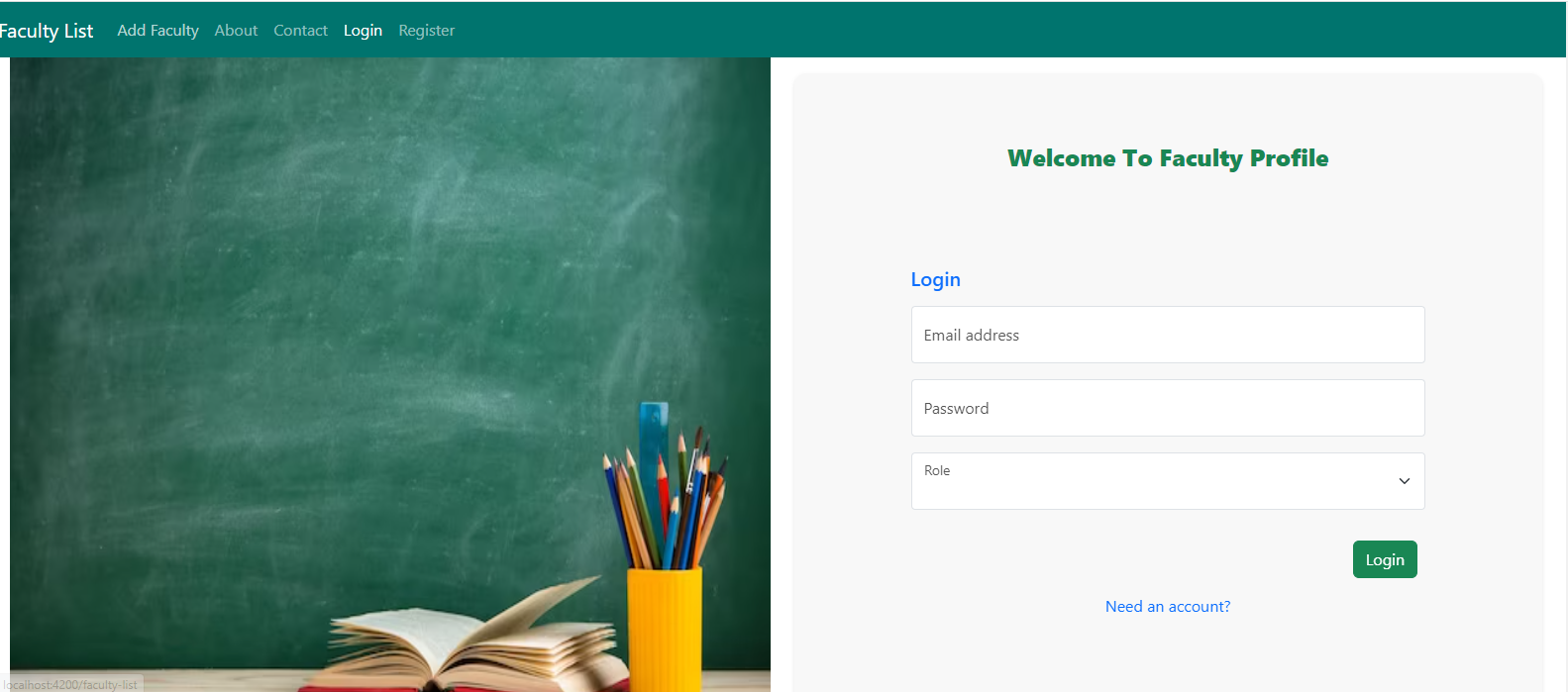
# 8.1 CONCLUSION

In conclusion, faculty profile management is a crucial system for educational institutions to efficiently and effectively manage the profiles of their faculty members. The successful implementation of such a system can lead to several benefits.

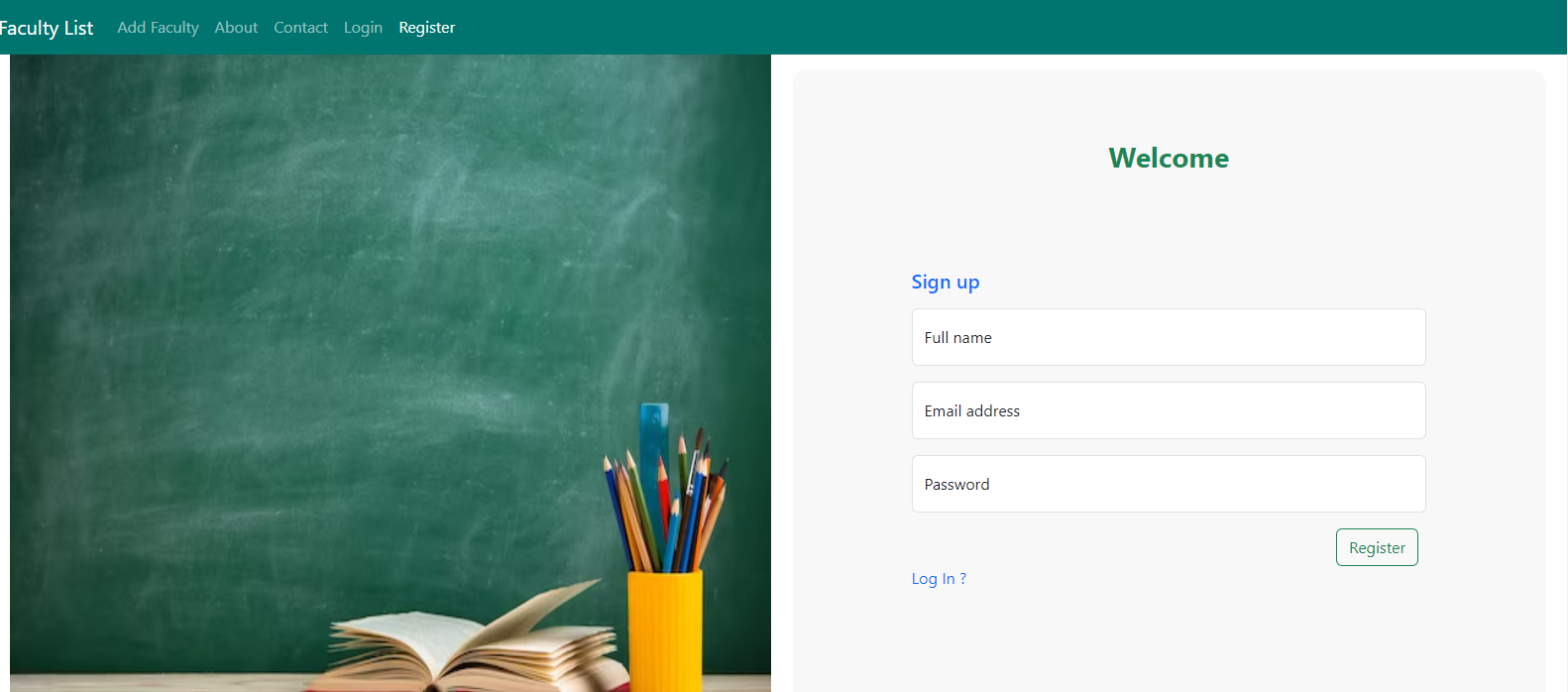
faculty profile management system is an indispensable tool for educational institutions seeking to streamline faculty information, enhance collaboration, and make data-driven decisions. The successful implementation of such a system requires careful planning, consideration of user requirements, and adherence to best practices in software development and data management. By prioritizing usability, security, and scalability, institutions can leverage this system to optimize faculty management and foster academic excellence.

# 8.2 SCREENSHOTS

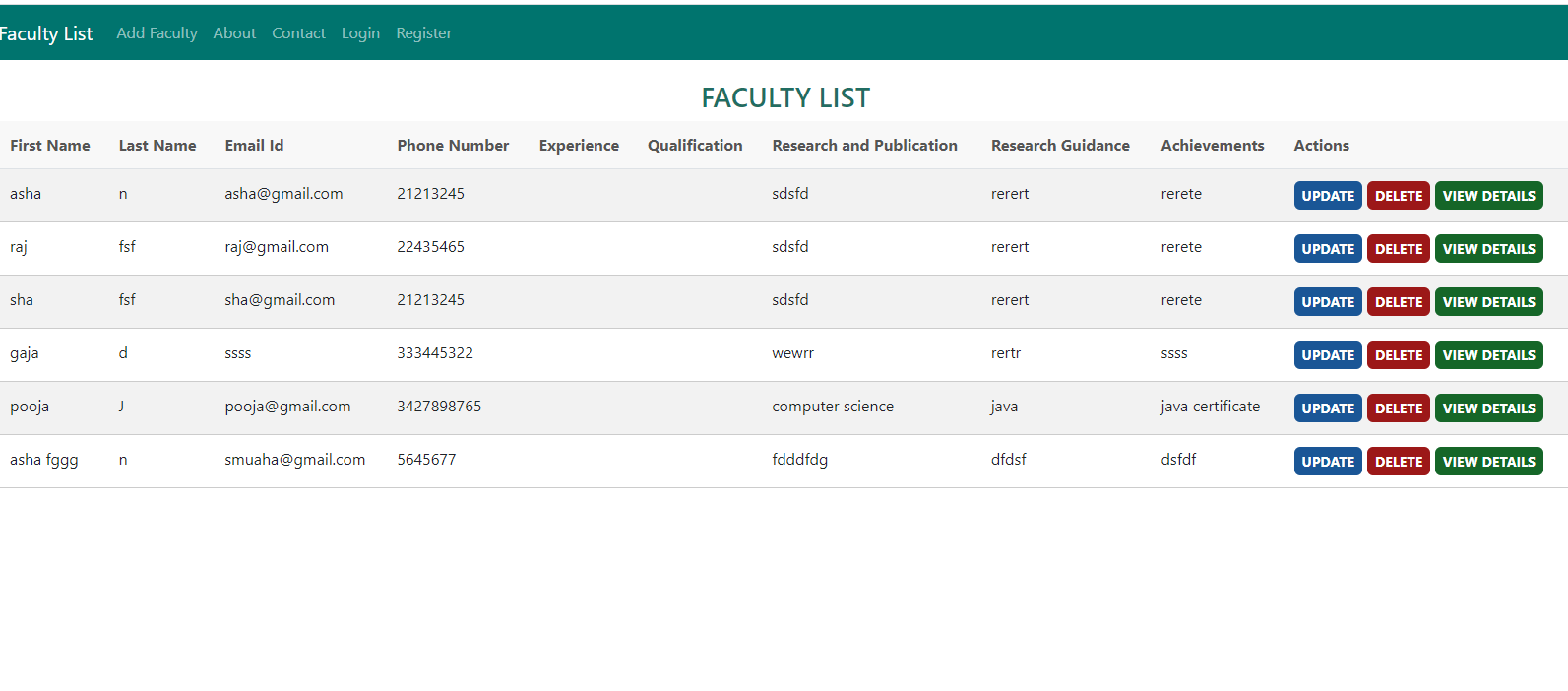
## Login Page:



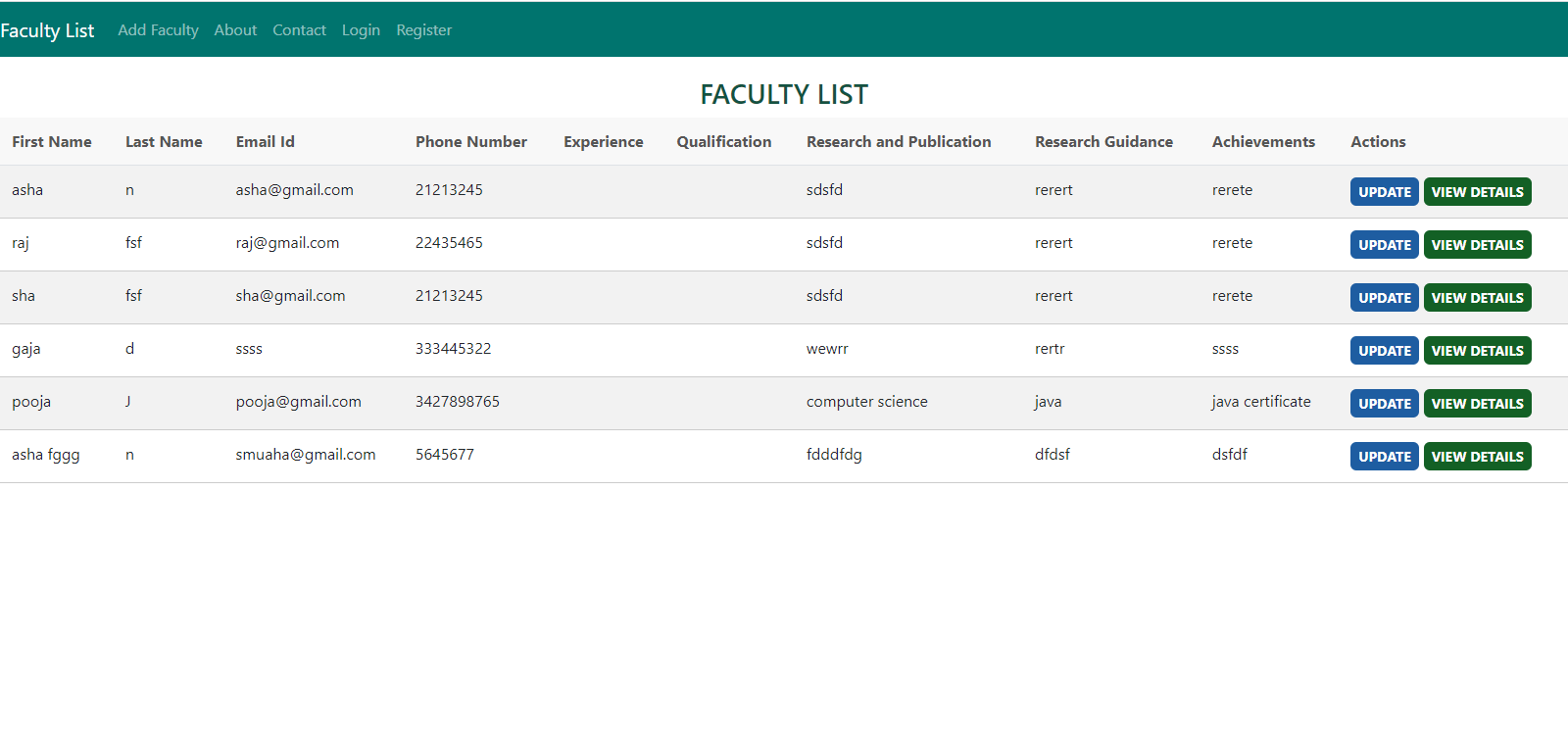
## Registration Page:



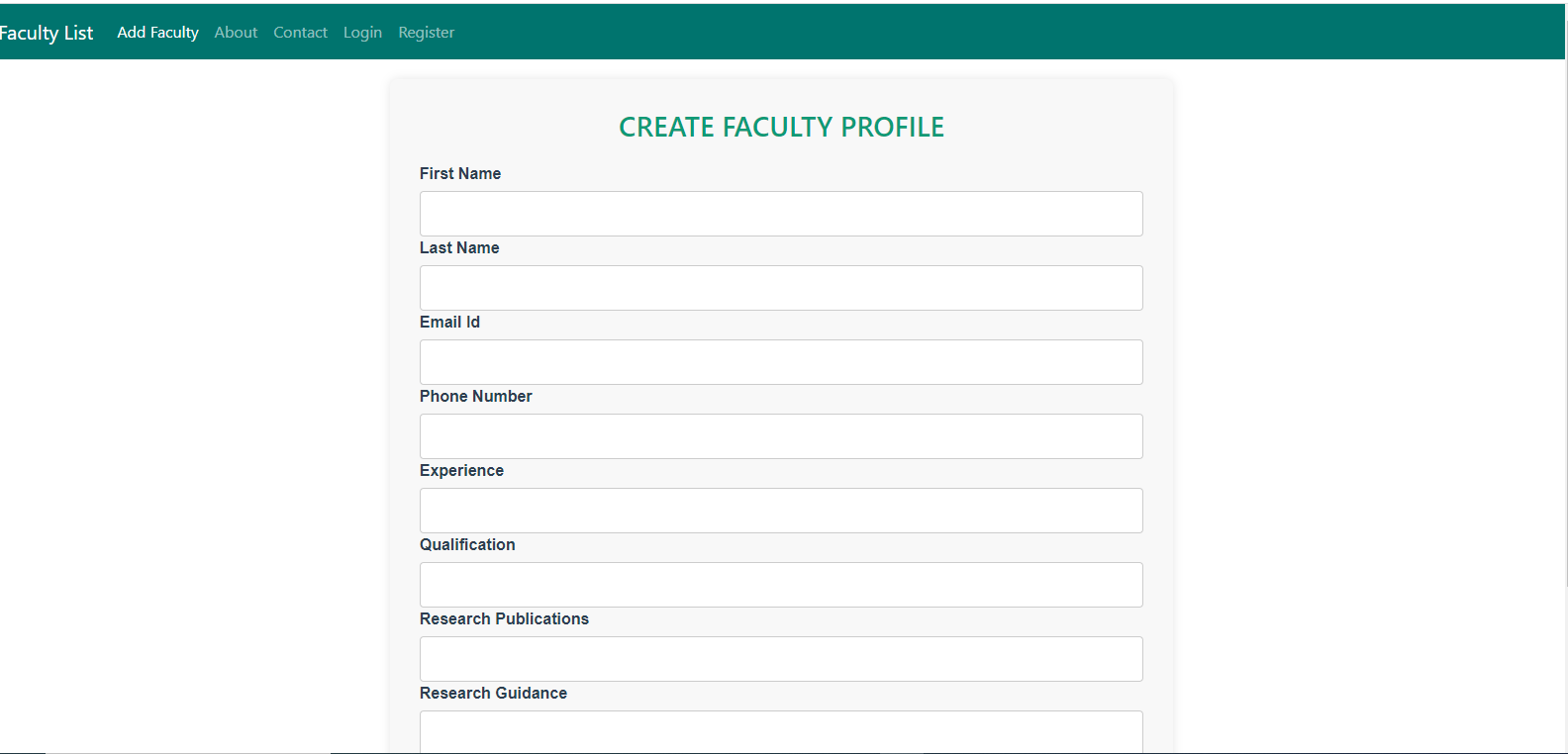
## Admin Dashboard:



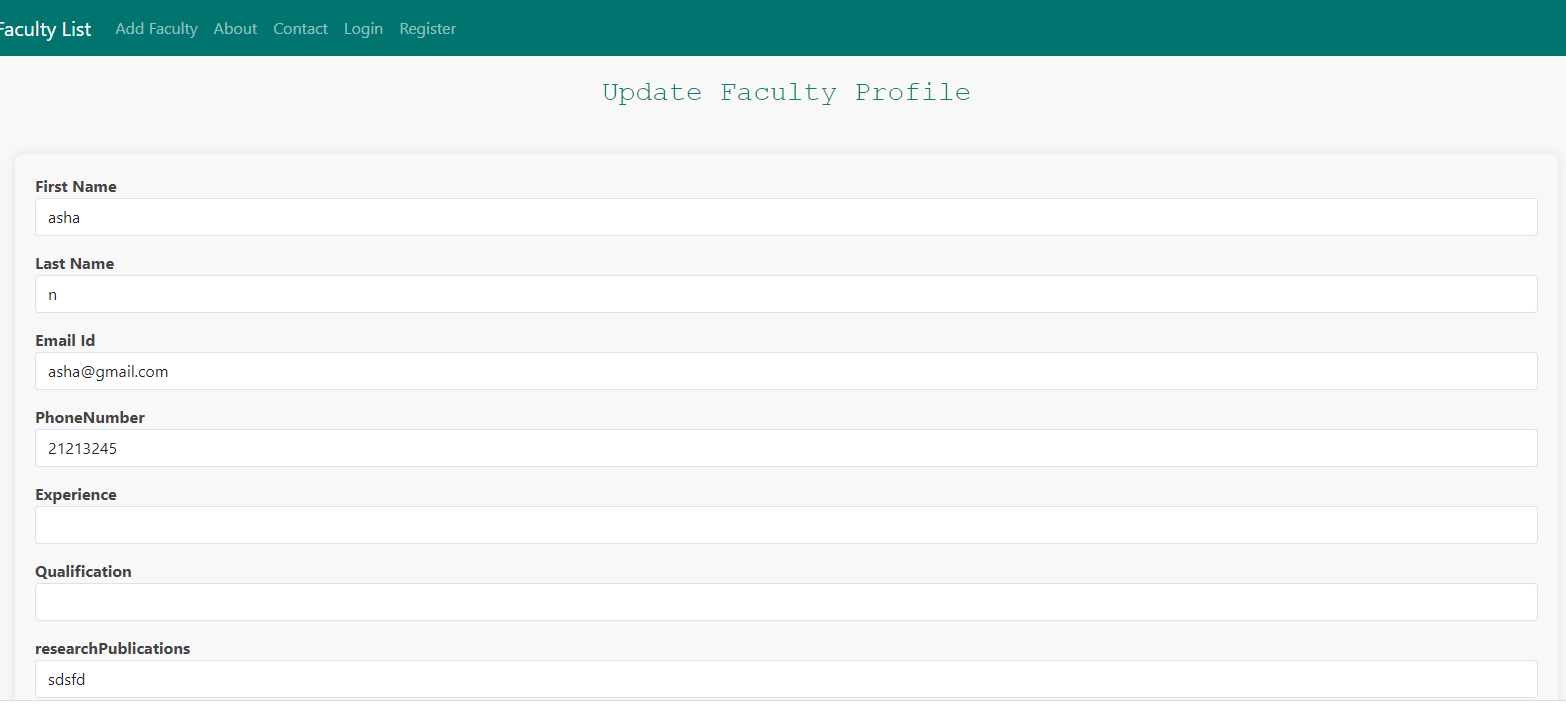
## User Dashboard:



## Add Faculty Page:



## Update Faculty Details:



Faculty details page:

