MANIPAL INSTITUTE OF TECHNOLOGY
MANIPAL
(A constituent unit of MAHE, Manipal)

FINTECH LAB MINI PROJECT

Problem Statement-In an organization there is hierarchy when approval process is required it should go to the immediate superior manager if the manager is not available then to it should go to the managers superior and so on

SUBMITTED BY-

1.Riddhi Tanwani (230958162)

2.Cyrus Tinny (230958017)

3.P Pranathi (230958016)

4.Isha Ranjan (230958034)

ACKNOWLEDGMENT

We would like to express my heartfelt gratitude to the faculty members of Fintech Lab , Computer Science and financial technology branch at **Manipal Institute of Technology, Manipal** for their constant support, encouragement, and valuable insights throughout the duration of this project.

A special thanks to the **lab in-charges and FIS employees** who conducted the practical sessions and provided us with a productive learning environment. Their timely guidance and the technical infrastructure available in the labs played a significant role in the successful implementation of this project.

We are also thankful to my fellow classmates and peers for their continuous support, collaboration, and motivation throughout this journey. Their feedback and discussions helped enhance the quality and scope of the project.

This project, *ApproveX – Approval Request System*, would not have been possible without the collective support of all the individuals who contributed in various ways. We are truly grateful for this enriching experience.

INDEX

1. ABSTRACT:

In large organizations, approval workflows often involve multiple hierarchical levels, leading to delays and inefficiencies when managed manually. ApproveX is a web-based Approval Request System designed to automate and streamline the approval process by leveraging organizational hierarchy. The system ensures that every request is routed to the immediate superior of the requester and, in case of unavailability, automatically escalates to the next level.

The application is built using Angular for the frontend, Spring Boot for the backend, and MySQL as the database. It features a clean and responsive user interface, RESTful API integration, role-based access control, and real-time tracking of approval statuses. ApproveX enhances transparency, accountability, and efficiency in handling approvals, making it a scalable solution for modern organizations.

2. INTRODUCTION

In any structured organization, decision-making and resource allocation often require formal approvals that follow a chain of command. Traditional approval processes are typically manual, time-consuming, and prone to delays—especially when a manager is unavailable to take timely action. These inefficiencies can disrupt workflows, affect productivity, and hinder effective communication.

ApproveX is a smart, full-stack web application developed to automate and optimize the approval workflow within an organization. It leverages the existing organizational hierarchy to ensure that approval requests are

sent to the appropriate manager and, in case of their unavailability, escalated automatically to the next superior authority. This eliminates bottlenecks and ensures requests are processed without delay.

The system is designed with a focus on user-friendliness, real-time tracking, and secure data handling. It provides distinct interfaces for employees and managers, enabling seamless request creation, approval management, and status updates. By combining technologies like Angular, Spring Boot, and MySQL, ApproveX delivers a robust and scalable solution that addresses the key challenges in conventional approval systems.

This report presents the motivation behind the project, its architectural design, system modules, implementation details, and the key outcomes achieved through ApproveX.

3. OBJECTIVES

The primary goal of the **ApproveX – Approval Request System** is to streamline and automate the approval workflow within an organization by leveraging its hierarchical structure. The specific objectives of the project are as follows:

1. To design and implement a hierarchical request routing system that sends approval requests to the immediate superior and escalates them when necessary.

2. To develop a user-friendly web interface for both requesters and managers using Angular, ensuring a smooth and intuitive user experience.

3. To build a secure and scalable backend
   using Spring Boot and MySQL that supports RESTful APIs for data handling and business logic.

4. To implement role-based access control
   allowing different functionalities and views based on user roles (employee, manager, admin).

5. To ensure real-time tracking and transparency
   so users can monitor the status of their requests throughout the approval process.

6. To handle validations, error management, and edge cases
   to ensure robust and reliable system behavior under all scenarios.

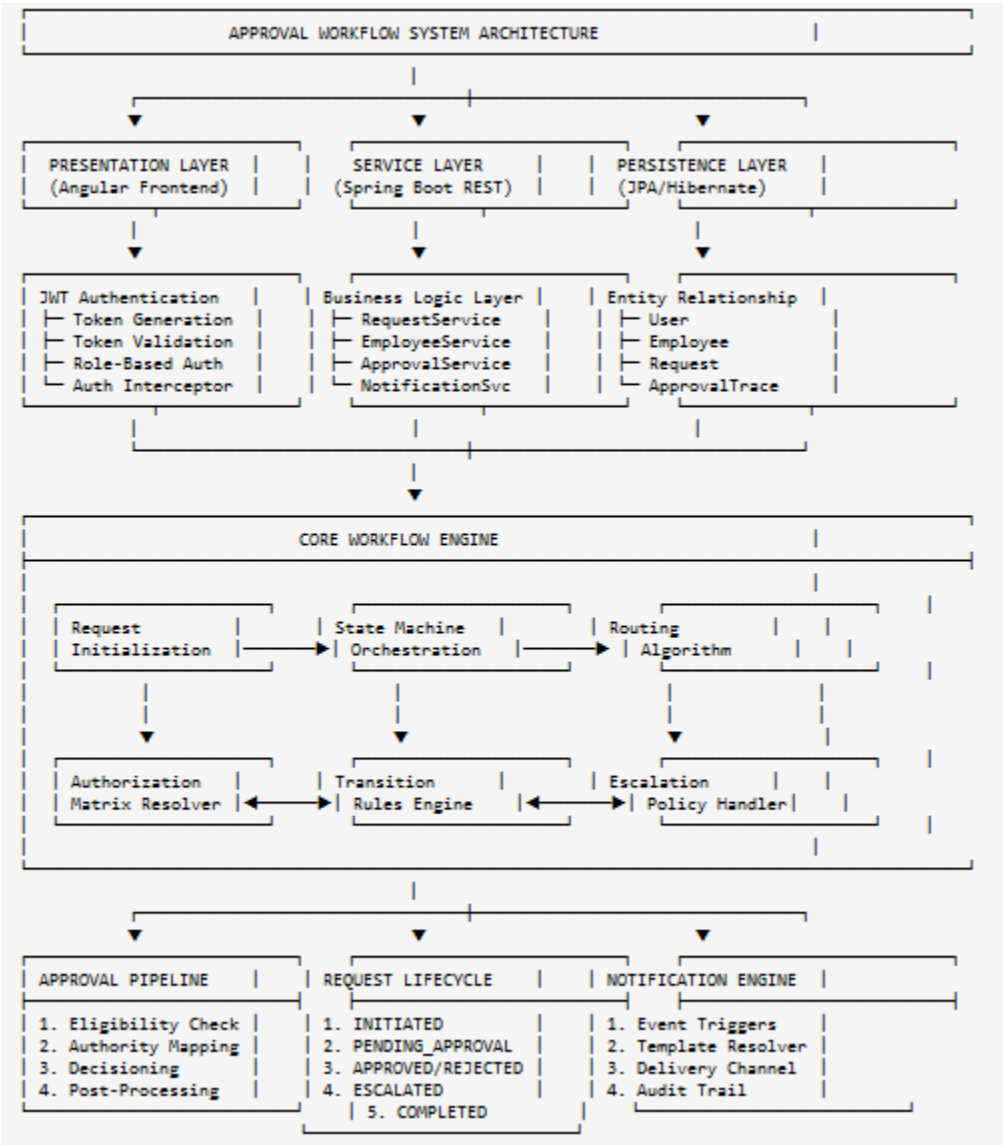
7. To thoroughly test API endpoints using Postman
   and validate proper integration between frontend and backend components.

4. TECHNOLOGY STACK

- **Frontend:** Angular
- **Backend:** Spring Boot
- **Database:** MySQL
- **Tools:** Postman, Eclipse IDE,VSCode, Git
- **Languages:** Java

5. SYSTEM ARCHITECTURE

- **Frontend** communicates via REST API with **Backend**.
- **Backend** implements service and controller layers with validation and exception handling.
- **Database** stores user, manager, and request data.



6. BACKEND IMPLEMENTATION (SPRING BOOT)

The backend of ApproveX is developed using Spring Boot, a robust Java-based framework that simplifies the development of production-grade applications. It handles business logic, data persistence, authentication, and communication with the frontend through RESTful APIs.

1. Project Setup

- The backend codebase is structured as a Maven project.By default, the application starts on port 8080 and serves secured REST APIs for all major functionalities.
- MySQL – As the primary relational database
- Maven – For project management and dependency handling
- Role-Based Access: Users are categorized into three roles: EMPLOYEE, MANAGER, and SUPERVISOR, each with defined permissions and dashboards.
- Hierarchy Routing: Approval requests are routed based on the organizational hierarchy. If a manager is unavailable, the request is escalated up the hierarchy automatically.
- JWT Authentication: Ensures secure and stateless authentication using access tokens. All endpoints are protected and require a valid JWT token.

2. Entity Relationships

The core entities include:

- Employee: Stores details such as name, role, availability, and supervisor linkage.

- ApprovalRequest: Captures approval details like requester, approver, description, status, and timestamps.
- A one-to-many relationship between supervisor and employees
- A many-to-one relationship between approval requests and both requester and approver

3. Core API Endpoints

Below are some critical RESTful endpoints:

Authentication-

- POST /api/auth/login – Logs in a user and returns a JWT token

Employee APIs-

- GET /api/employees – Retrieve all employees
- POST /api/employees – Create a new employee
- PUT /api/employees/{id}/supervisor – Assign or change a supervisor
- PUT /api/employees/{id}/toggle-availability – Change availability status

Approval Request APIs-

- POST /api/approvals – Submit a new request
- PUT /api/approvals/{id}/approve – Approve a request
- PUT /api/approvals/{id}/reject – Reject a request
- PUT /api/approvals/{id}/escalate – Escalate to next level

All APIs are secured and require a valid JWT token in the Authorization header.

4. Database Management

- MySQL is used to store employee and request data.
- Credentials are configured in the application.properties file.
- All tables are automatically generated and managed using JPA and Hibernate.

5. Error Handling & Validations

- All endpoints include custom exception handling using @ControllerAdvice.
- Validation annotations such as @NotBlank, @Email, etc., are used for request DTOs to prevent bad data.

7. FRONTEND IMPLEMENTATION (ANGULAR)

The frontend of **ApproveX** is built using **Angular**, a powerful TypeScript-based framework widely used for developing dynamic, single-page web applications. The frontend provides a responsive and intuitive interface that interacts with the backend REST APIs and handles different roles like **Employee**, **Manager**, and **Supervisor** seamlessly.

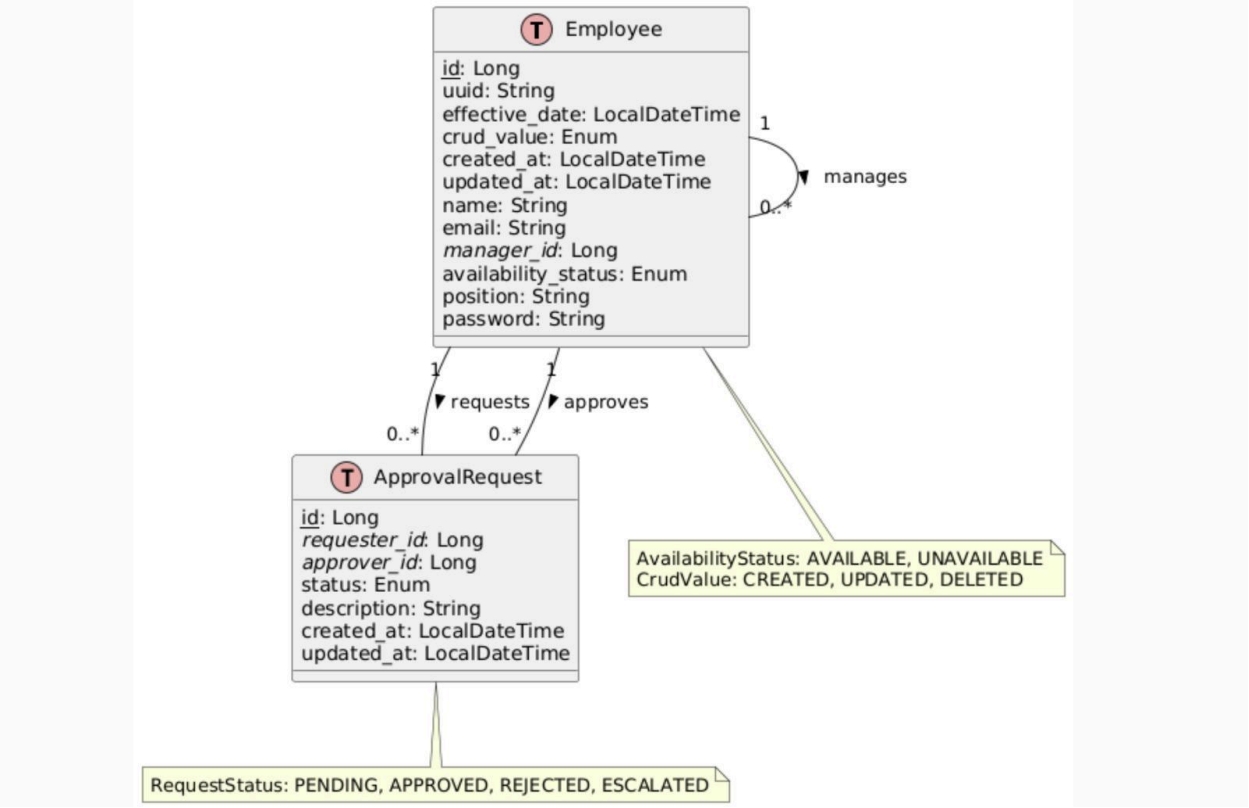
1. Project Setup

The Angular frontend is placed inside the Spring Boot project's static resources directory:

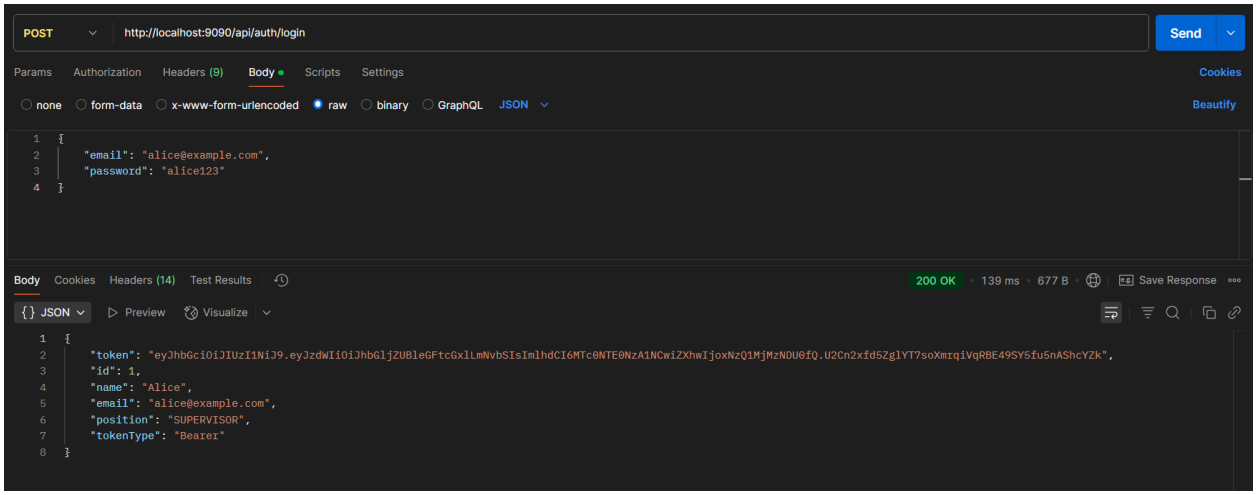
2. Technologies Used

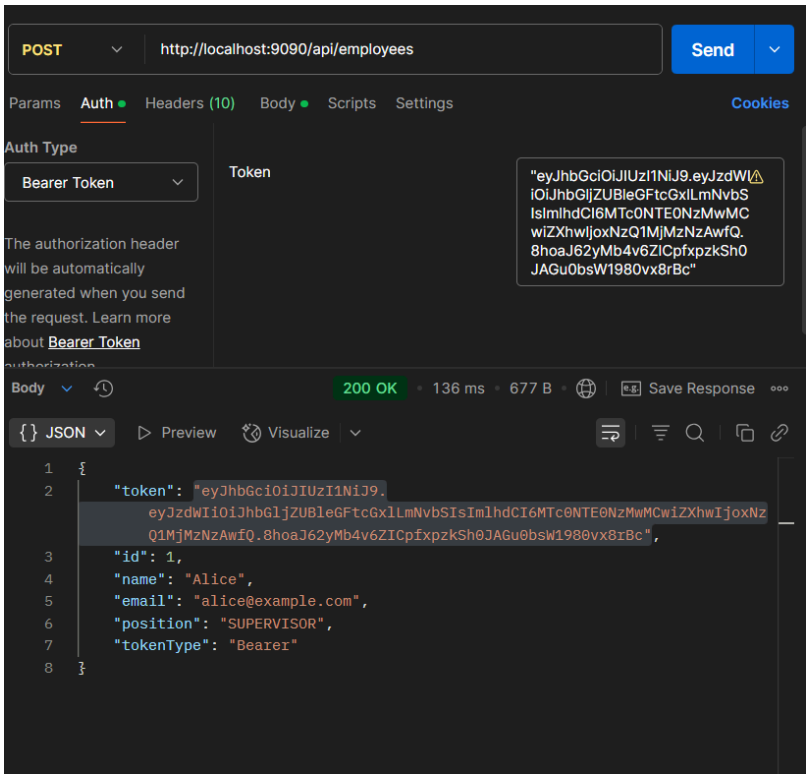
- Angular (v14+) – For building the UI
- TypeScript – For frontend logic
- HTML5 & SCSS – For structure and styling
- Angular Router – For handling navigation between pages
- Angular Services & HTTP Client – For consuming backend APIs
- Reactive Forms – For form handling and validations
- Bootstrap / Material UI (if used) – For enhanced UI components (based on your actual usage)

8. ENTITY RELATIONSHIP DIAGRAM



9. API TESTING USING POSTMAN





POST http://localhost:9090/api/employees Send

Params Auth Headers (10) Body Scripts Settings Cookies

raw JSON Beautify

```
{
    "name": "John Doe",
    "email": "john@example.com",
    "position": "EMPLOYEE",
    "password": "john123"
}
```

Body 201 Created 252 ms 821 B Save Response

JSON Preview Visualize


```
{
    "uuid": "e20a8d5a-b900-4b77-b60d-9296ce91af1c",
    "effectiveDate": "2025-04-20T16:39:40.9685797",
    "crudValue": "CREATED",
    "createdAt": "2025-04-20T16:39:41.054727",
    "updatedAt": "2025-04-20T16:39:41.054727",
    "id": 4,
    "name": "John Doe",
    "email": "john@example.com",
    "manager": null,
    "availabilityStatus": "AVAILABLE",
    "position": "EMPLOYEE",
    "password": "$2a$10$Be5AsIR3K.Bbs4VwyYmb6OVhs7kB2aIAuuqvyeASiSsSA2C1D3Rgq"
}
```

GET http://localhost:9090/api/employees Send

Params Auth Headers (8) Body Scripts Settings Cookies

Auth Type

Bearer Token

Token eyJhbGciOiJIUzI1NiJ9.eyJzdW...

The authorization header will be automatically generated when you send the request. Learn more about Bearer Token authorization.

Body 200 OK 48 ms 3.07 KB Save Response

JSON Preview Visualize


```
[
    {
        "uuid": "9ada0aa2-716e-4a44-9952-ed37650db1d8",
        "effectiveDate": "2025-04-20T16:33:52.920914",
        "crudValue": "CREATED",
        "createdAt": "2025-04-20T16:33:53.025426",
        "updatedAt": "2025-04-20T16:33:53.025426",
        "id": 1,
        "name": "Alice",
        "email": "alice@example.com",
        "manager": null,
        "availabilityStatus": "AVAILABLE",
        "position": "SUPERVISOR",
        "password": "$2a$10$u3/pmY7H9F0ah40LmDrEX.IScvdA32icP.HluP5/OAcaUQZ1L7EtO"
    },
    {
        "uuid": "dfa1c522-9efb-46d0-9e4b-1ae29289c8ff",
        "effectiveDate": "2025-04-20T16:33:53.062773"
```

PUT http://localhost:9090/api/employees/3 Send

Params Auth Headers (10) Body Scripts Settings Cookies

raw JSON Beautify

```
{
    "name": "Charlie Brown",
    "email": "charlie@example.com",
    "position": "EMPLOYEE",
    "password": "newpassword123"
}
```

Body 200 OK 213 ms 823 B Save Response

{} JSON Preview Visualize


```
{
    "uuid": "71a0d7a5-a8a7-45df-8f82-9192da007e4f",
    "effectiveDate": "2025-04-20T16:33:53.131321",
    "crudValue": "CREATED",
    "createdAt": "2025-04-20T16:33:53.196384",
    "updatedAt": "2025-04-20T16:41:40.557739",
    "id": 3,
    "name": "Charlie Brown",
    "email": "charlie@example.com",
    "manager": null,
    "availabilityStatus": "AVAILABLE",
    "position": "EMPLOYEE",
    "password": "$2a$10$ez8CqmUmyhsdEpBuIb980.FWJRZXHs3.k24v68tWQxzees3e3DPT."
}
```

POST http://localhost:9090/api/auth/login Send

Params Auth Headers (9) Body Scripts Settings Cookies

raw JSON Beautify

```
{
    "email": "charlie@example.com",
    "password": "newpassword123"
}
```

Body 200 OK 119 ms 687 B Save Response

{} JSON Preview Visualize


```
{
    "token": "eyJhbGciOiJIUzI1NiJ9.
        eyJzdWIiOiJjaGFybGllQGV4YW1wbGUuY29tIiwiaWF0IjoxNzQ1MTQ3Njc2LCJleHAiOj
        E3NDUyMzQwNzZ9.M7RJjSFLUBFp1imiKLyc3mKDn6S1HZVtkvbnKmiMOCg",
    "id": 3,
    "name": "Charlie Brown",
    "email": "charlie@example.com",
    "position": "EMPLOYEE",
    "tokenType": "Bearer"
}
```

POST http://localhost:9090/api/approvals Send

Params Auth Headers (10) Body Scripts Settings Cookies

Auth Type

Bearer Token

Token eyJhbGciOiJIUzI1NiJ9.eyJzdW...

The authorization header will be automatically generated when you send the request. Learn more about Bearer Token authorization.

Body 200 OK 119 ms 687 B Save Response

JSON Preview Visualize


```
{
    "token": "eyJhbGciOiJIUzI1NiJ9.
        eyJzdWIiOiJjaGFybGllQGV4YW1wbGUuY29tIiwiaWF0IjoxNzQ1MTQ3Njc2LCJleHAiOj
        E3NDUyMzQwNzZ9.M7RJjSFLUBFp1imiKLyc3mKDn6S1HZVtkvbnKmiMOCg",
    "id": 3,
    "name": "Charlie Brown",
    "email": "charlie@example.com",
    "position": "EMPLOYEE",
    "tokenType": "Bearer"
}
```

POST http://localhost:9090/api/approvals

Send

Params Auth Headers (10) Body Scripts Settings Cookies

raw JSON Beautify

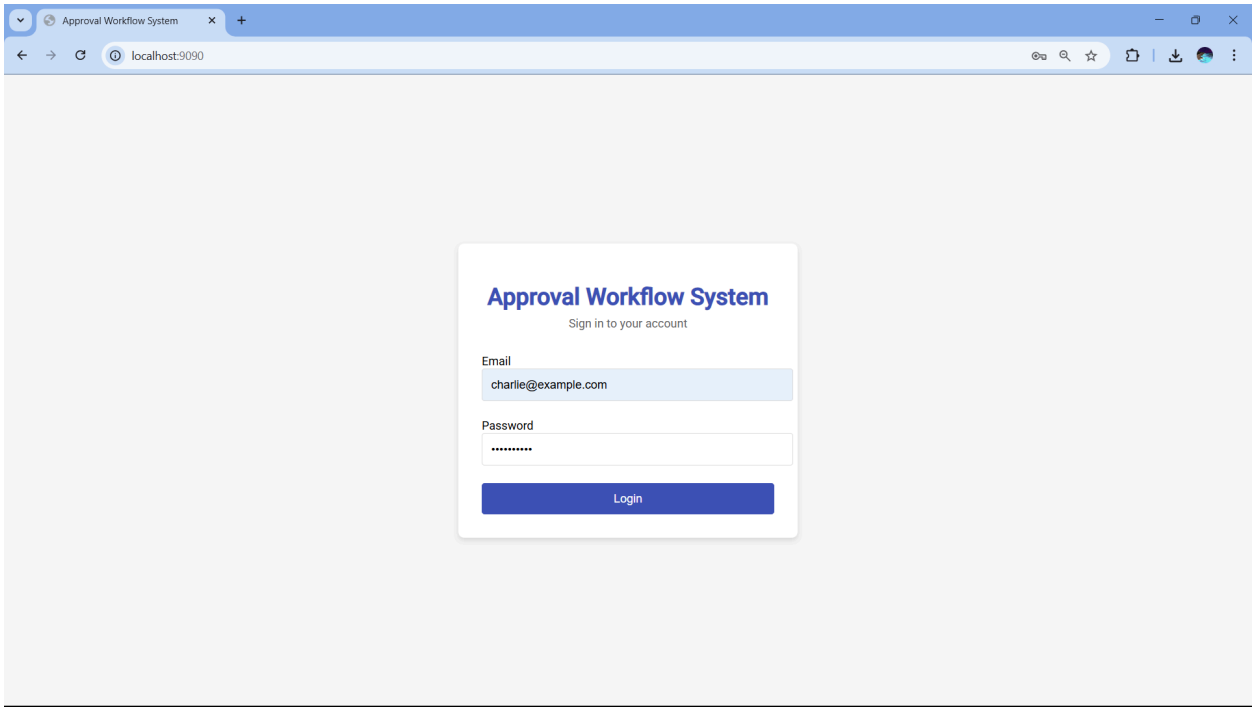
```
{
    "requesterId": 3,
    "description": "Request for vacation leave from June 1-5"
}
```

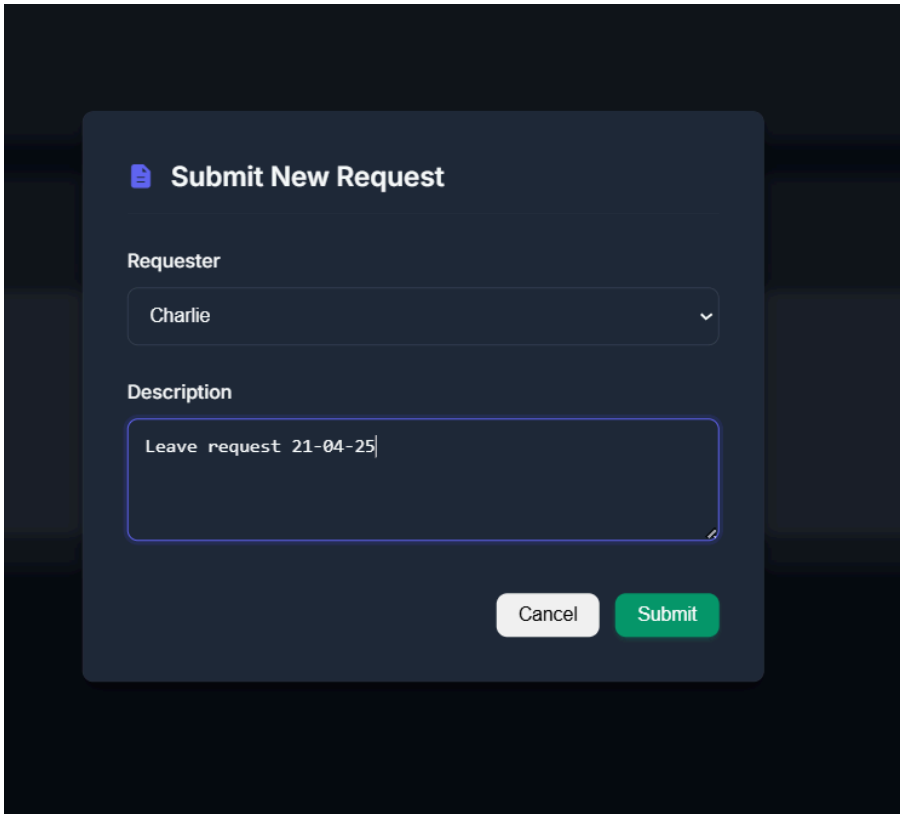
Body 201 Created 51 ms 666 B Save Response

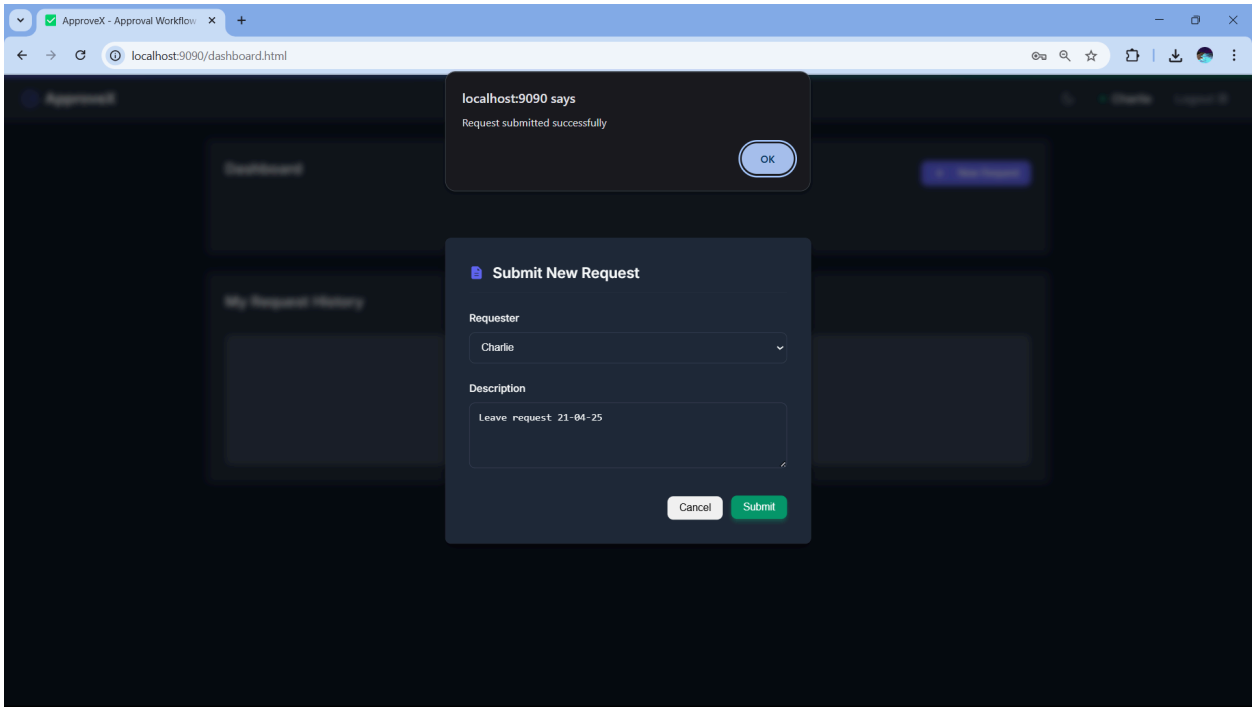
{} JSON Preview Visualize

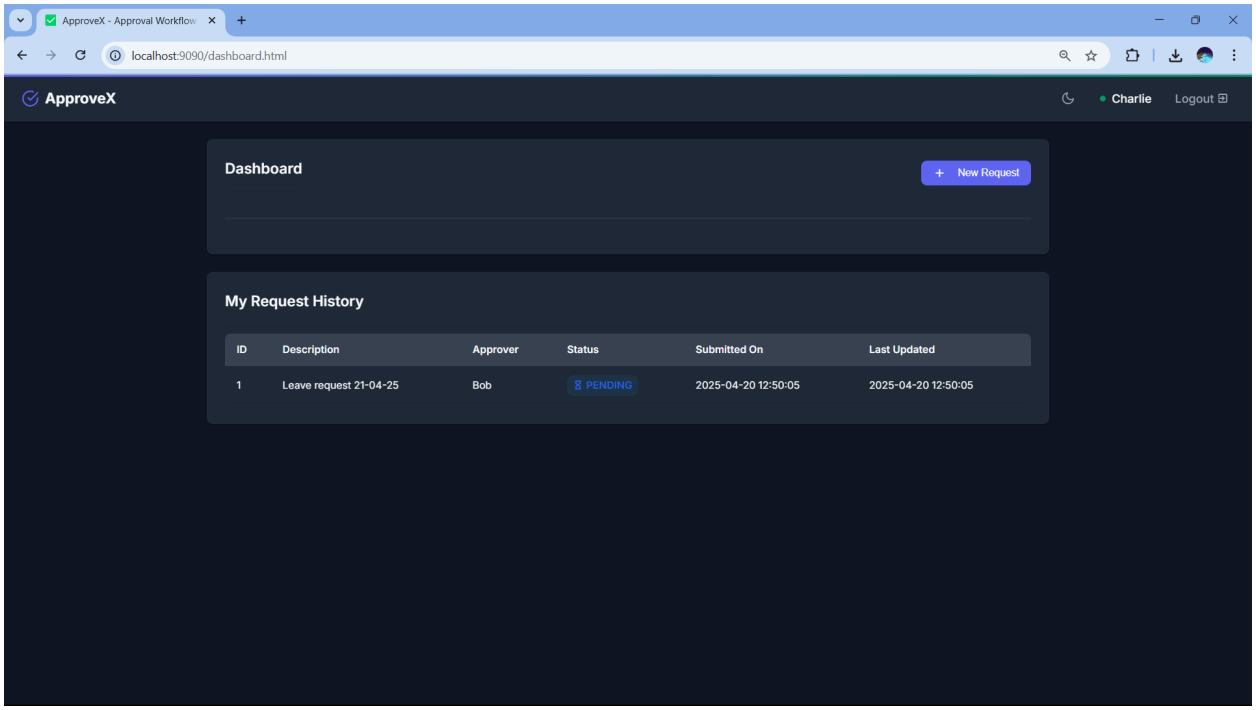
```
{
    "id": 1,
    "requesterId": 3,
    "approverId": null,
    "status": "PENDING",
    "description": "Request for vacation leave from June 1-5",
    "createdAt": "2025-04-20 16:48:34",
    "updatedAt": "2025-04-20 16:48:34",
    "requesterName": "Charlie Brown",
    "approverName": null
}
```

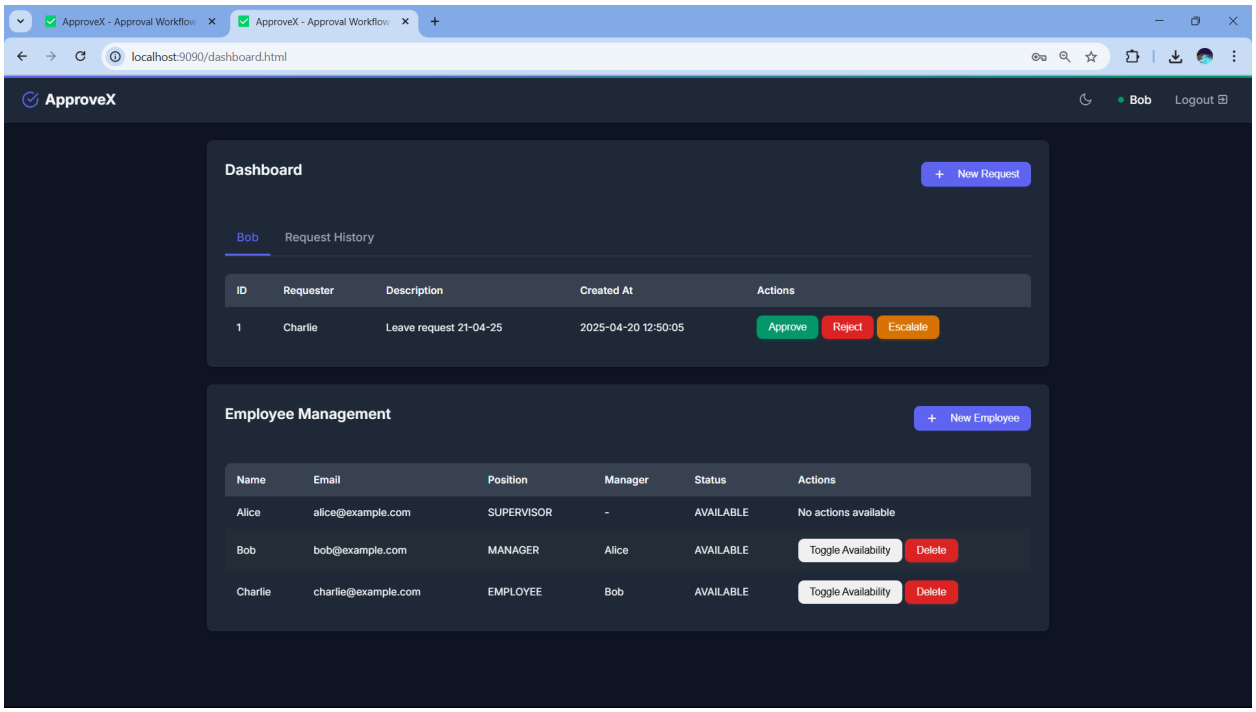
10. UI SCREENS

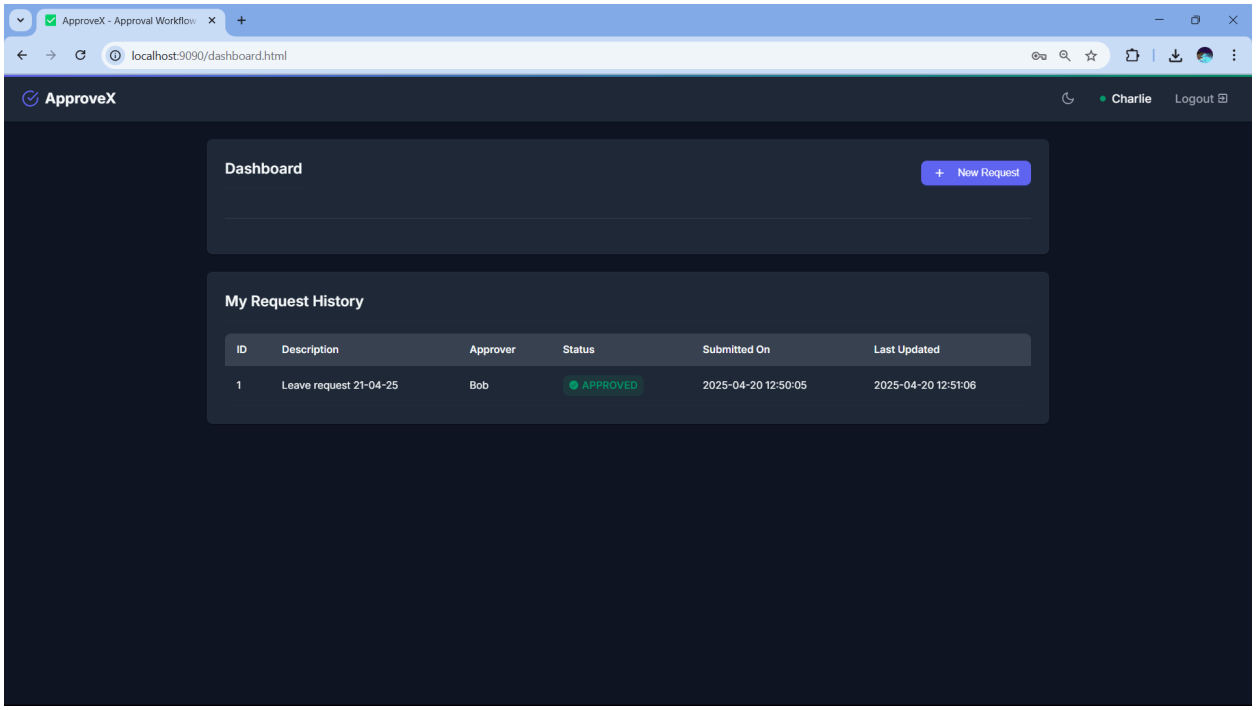


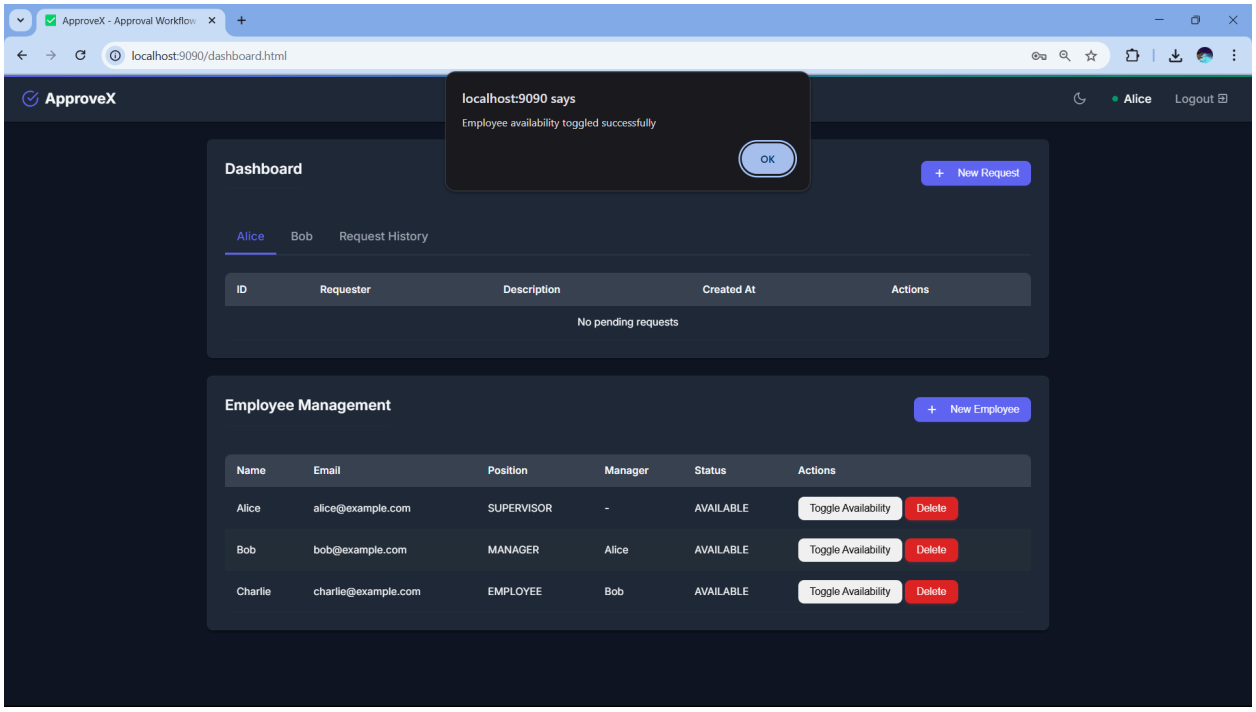

Submit New Request
Requester
Charlie
Description
Leave request 21-04-25
Cancel
Submit

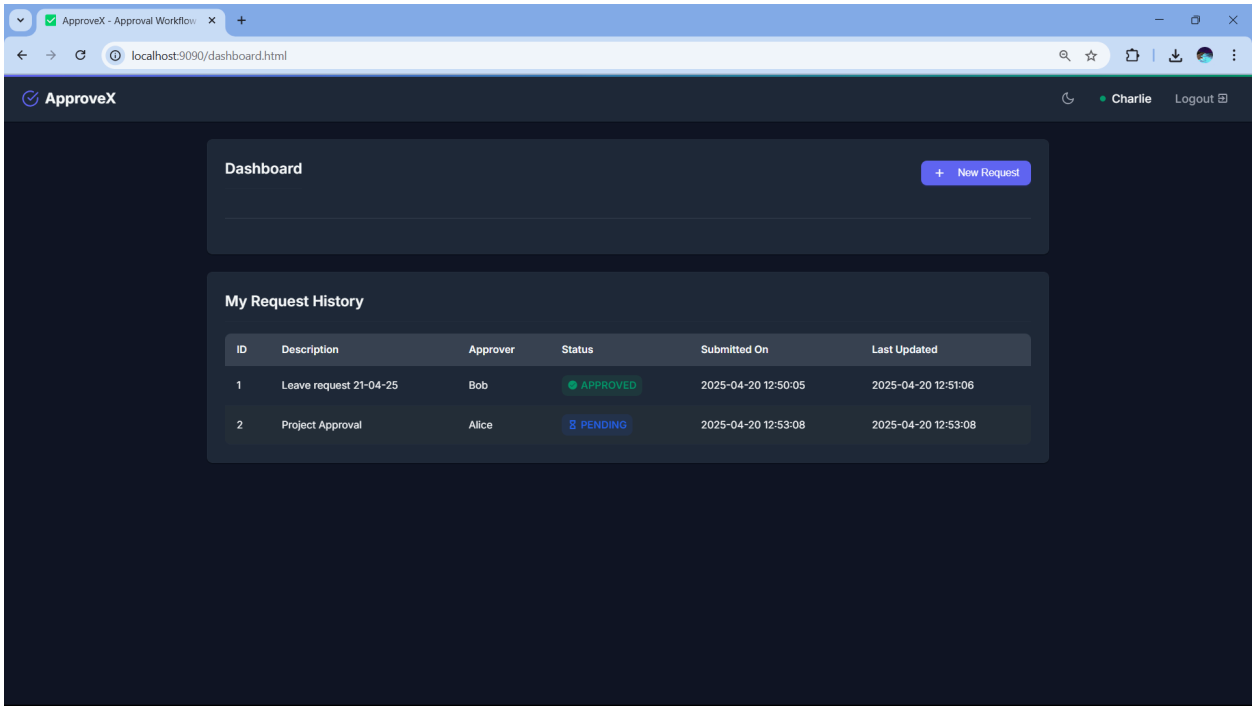

ApproveX - Approval Workflow
localhost:9090/dashboard.html
localhost:9090 says
Request submitted successfully
OK
Submit New Request
Requester
Charlie
Description
Leave request 21-04-25
Cancel
Submit

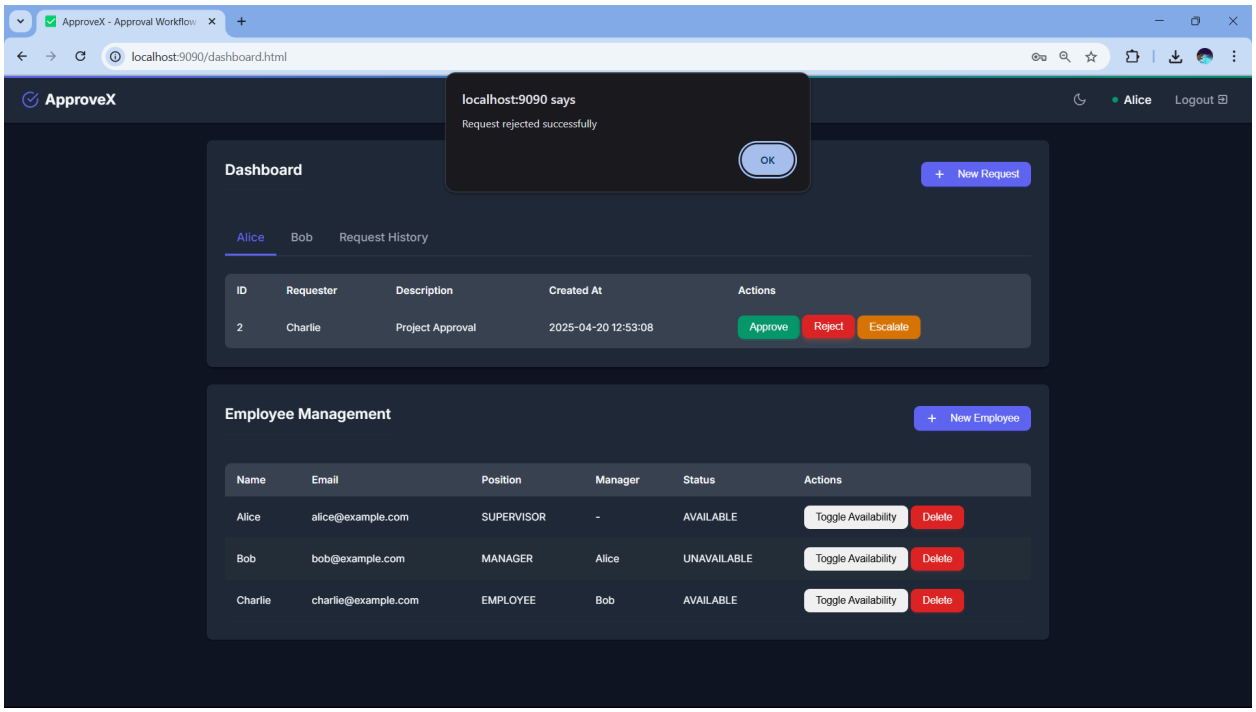

ApproveX - Approval Workflow
localhost:9090/dashboard.html
ApproveX
Charlie
Logout
Dashboard
New Request
My Request History
ID
Description
Approver
Status
Submitted On
Last Updated
1
Leave request 21-04-25
Bob
PENDING
2025-04-20 12:50:05
2025-04-20 12:50:05


ApproveX - Approval Workflow
ApproveX - Approval Workflow
localhost:9090/dashboard.html
ApproveX
Bob
Logout
Dashboard
New Request
Bob
Request History
ID
Requester
Description
Created At
Actions
1
Charlie
Leave request 21-04-25
2025-04-20 12:50:05
Approve
Reject
Escalate
Employee Management
New Employee
Name
Email
Position
Manager
Status
Actions
Alice
alice@example.com
SUPERVISOR
-
AVAILABLE
No actions available
Bob
bob@example.com
MANAGER
Alice
AVAILABLE
Toggle Availability
Delete
Charlie
charlie@example.com
EMPLOYEE
Bob
AVAILABLE
Toggle Availability
Delete


ApproveX - Approval Workflow
localhost:9090/dashboard.html
ApproveX
Charlie
Logout
Dashboard
New Request
My Request History
ID
Description
Approver
Status
Submitted On
Last Updated
1
Leave request 21-04-25
Bob
APPROVED
2025-04-20 12:50:05
2025-04-20 12:51:06


ApproveX - Approval Workflow
localhost:9090/dashboard.html
ApproveX
Alice
Logout
localhost:9090 says
Employee availability toggled successfully
OK
Dashboard
New Request
Alice
Bob
Request History
ID
Requester
Description
Created At
Actions
No pending requests
Employee Management
New Employee
Name
Email
Position
Manager
Status
Actions
Alice
alice@example.com
SUPERVISOR
-
AVAILABLE
Toggle Availability
Delete
Bob
bob@example.com
MANAGER
Alice
AVAILABLE
Toggle Availability
Delete
Charlie
charlie@example.com
EMPLOYEE
Bob
AVAILABLE
Toggle Availability
Delete


ApproveX - Approval Workflow
localhost:9090/dashboard.html
ApproveX
Charlie
Logout
Dashboard
New Request
My Request History
ID
Description
Approver
Status
Submitted On
Last Updated
1
Leave request 21-04-25
Bob
APPROVED
2025-04-20 12:50:05
2025-04-20 12:51:06
2
Project Approval
Alice
PENDING
2025-04-20 12:53:08
2025-04-20 12:53:08


ApproveX - Approval Workflow
localhost:9090/dashboard.html
ApproveX
Alice
Logout
localhost:9090 says
Request rejected successfully
OK
Dashboard
New Request
Alice
Bob
Request History
ID
Requester
Description
Created At
Actions
2
Charlie
Project Approval
2025-04-20 12:53:08
Approve
Reject
Escalate
Employee Management
New Employee
Name
Email
Position
Manager
Status
Actions
Alice
alice@example.com
SUPERVISOR
-
AVAILABLE
Toggle Availability
Delete
Bob
bob@example.com
MANAGER
Alice
UNAVAILABLE
Toggle Availability
Delete
Charlie
charlie@example.com
EMPLOYEE
Bob
AVAILABLE
Toggle Availability
Delete

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

created_at	effective_date	id	manager_id	updated_at	email	name	password
2025-04-20 12:47:51.359649	2025-04-20 12:47:51.261053	1	NULL	2025-04-20 12:47:51.359649	alice@example.com	Alice	$2a$10$FDuO4sOhTJQPBbdQ7lZwtOQv4mKDH.
2025-04-20 12:47:51.455646	2025-04-20 12:47:51.388706	2	1	2025-04-20 12:52:29.013664	bob@example.com	Bob	$2a$10$Nn/hGbLqDDn7NXF/UYnNDegi7vU6buQ
2025-04-20 12:47:51.519552	2025-04-20 12:47:51.459849	3	2	2025-04-20 12:47:51.519552	charlie@example.com	Charlie	$2a$10$Ki6O5nqnmxBjiYYirqwOmuEuQJJFsbSy.
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Result Grid | Form Editor

employee 13 | Apply | Revert

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

approver_id	created_at	id	requester_id	updated_at	description	status
2	2025-04-20 12:50:05.763897	1	3	2025-04-20 12:51:06.639897	Leave request 21-04-25	APPROVED
1	2025-04-20 12:53:08.565429	2	3	2025-04-20 12:53:37.199469	Project Approval	REJECTED
NULL	NULL	NULL	NULL	NULL	NULL	NULL

Result Grid | Form Editor

approval_request14 | Apply | Revert

11. WORKFLOW & APPROVAL HIERARCHY

The **ApproveX** application is designed to streamline the approval request process in an organization by automating the routing of approvals through a hierarchical structure. The workflow ensures that requests are attended to efficiently, even in the absence of a specific manager, by escalating them to the next available superior.

1. **Login & Authentication**:
    - Users log in with their credentials.
    - Based on their role (EMPLOYEE, MANAGER, SUPERVISOR), they are redirected to their respective dashboards.

2. **Request Submission**:
    - An employee fills out the approval request form and submits it.

- The request includes details like requester name, description, and timestamp.

3. **Request Assignment**:

    - The system automatically assigns the request to the immediate supervisor/manager of the requester.

4. **Managerial Action**:

    - The assigned approver can Approve, Reject, or Escalate the request.
    - If the approver is unavailable, the request is escalated automatically to the next higher-level supervisor.

5. **Escalation Mechanism**:

    - The escalation continues upward in the hierarchy until a responsible and available manager takes action.
    - This prevents delays in the workflow due to unavailability of any one individual.

6. **Request Tracking**:

    - Both requester and approvers can track the status of requests (Pending, Approved, Rejected, Escalated) in real time.

7. **Admin Control**:

    - The admin can manage employees, update their availability, and assign/change supervisors.
    - This ensures that the hierarchy stays up to date and accurate.

Approval Hierarchy-

SUPERVISOR

↑

MANAGER

↑

EMPLOYEE

- **Employee**: Can submit requests and view request statuses.
- **Manager**: First-level approver for requests submitted by employees.
- **Supervisor**: Higher-level approver in case the manager is unavailable or escalates the request.
- **Admin**: Has control over the employee database and can configure the approval chain.

This dynamic and fail-safe routing system ensures that no request is left unattended, maintaining organizational efficiency and accountability.

12. CHALLENGES FACED

While developing **ApproveX**, we faced several practical challenges—both technical and conceptual—that taught us a lot during the process:

- **Handling Relationships in Spring Boot-**Setting up entity relationships was tricky, especially because one employee can be another's supervisor. Managing those self-referencing relationships without breaking the data flow or causing issues while saving records was a challenge, especially when dealing with cascading updates and JSON responses.

- **Integrating Frontend and Backend-**We ran into multiple issues while trying to connect the Angular frontend with the Spring Boot backend. CORS errors, mismatched endpoints, and token-related issues during login took up a lot of time until everything finally started working smoothly.

- **Authentication and Token Handling-**Getting JWT-based login to work properly and storing the token securely in Angular was new for us. Sometimes the token wouldn't attach to the request header properly, causing access to fail for protected routes.

- **Role-Based UI Rendering-**Making sure that the right components and data show up based on the logged-in user's role (EMPLOYEE, MANAGER, SUPERVISOR, ADMIN) was more complicated

than expected. We had to conditionally show/hide buttons, forms, and even routes based on the role, and it took time to set this up properly.

- **Understanding and Implementing the Approval Hierarchy Logic-**Initially, it was confusing to figure out how to implement a system where the approval request goes to the immediate manager and escalates upwards only if they're unavailable. Writing the logic to dynamically find the right next approver based on availability took a few tries and debugging.

13. FUTURE SCOPE

While ApproveX successfully automates the approval request process within an organization, there are several ways this system can be enhanced in the future:

- Email and SMS Notifications

  Integrating email or SMS alerts for request approvals, rejections, or escalations can keep users informed in real-time, especially when they are not logged into the system.

- Mobile App Integration

Developing a mobile version of ApproveX would allow users to submit and act on approvals from anywhere, making it more accessible and convenient.

- Advanced Reporting and Analytics

  Adding dashboards with analytics to show approval timelines, pending requests, and employee performance could help managers make better decisions.

- Calendar and Leave Integration

  Syncing the system with employee calendars or leave data would improve the availability logic by automatically identifying when a manager is on leave.

- Document Attachment and History Logs

  Allowing users to attach files with requests and maintaining a complete approval history log would add more transparency and accountability.

- Cloud Deployment & Scalability

  Hosting the application on platforms like AWS or Azure with scalable infrastructure would allow it to handle larger organizations and more concurrent users efficiently.

14. CONCLUSION

ApproveX successfully achieves the goal of automating and simplifying approval workflows. It enhances transparency and saves valuable organizational time, making it a scalable and efficient solution.

15.GITHUB AND PRESENTATION LINK

Presentation link

Github code

Video link

16.REFERENCES

1. Dumas, M., La Rosa, M., Mendling, J., & Reijers, H. A. (2018). *Fundamentals of Business Process Management*. Springer.

2. Workflow Patterns Initiative. Available at: https://www.workflowpatterns.com/

3. Johnson, C. (2017). *Spring Microservices in Action*. Manning Publications.

4. Baeldung – Spring Boot Tutorials. Available at: https://www.baeldung.com/

5. Freeman, A., & Sanderson, S. (2022). *Pro Angular 9*. Apress.

6. Angular Documentation. Available at: https://angular.io/docs

7. OAuth 2.0 and OpenID Connect – Auth0 Documentation. Available at: https://auth0.com/docs

8. Coronel, C., & Morris, S. (2016). *Database Systems: Design, Implementation, and Management*. Cengage Learning.

9. Postman Learning Center. Available at: https://learning.postman.com/

10. Amazon Web Services (AWS) Documentation. Available at: https://docs.aws.amazon.com/

11. Gartner, L. (2021). *Cloud Native Spring in Action*. Manning Publications.