FILE SERVER RBAC

About the Project

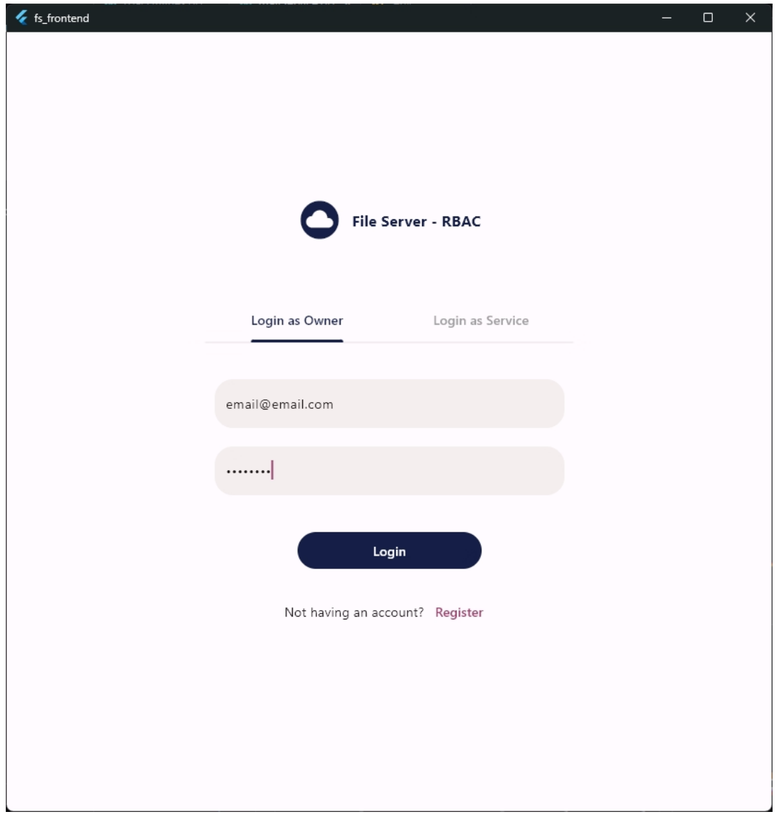
File Server with RBAC controls is an advanced system to enable users to create workspaces and it associated users and roles to manage. It enhances and organizes the way a user can access the files and folders with roles. RBAC is employed as a robust access control mechanism, allowing administrators to assign specific roles to users based on their responsibilities and permissions. This ensures a more granular control over file and directory access. The File Server with RBAC project thus contributes to a more secure, efficient, and organized data handling environment within an enterprise or networked system.

Features of FS-RBAC

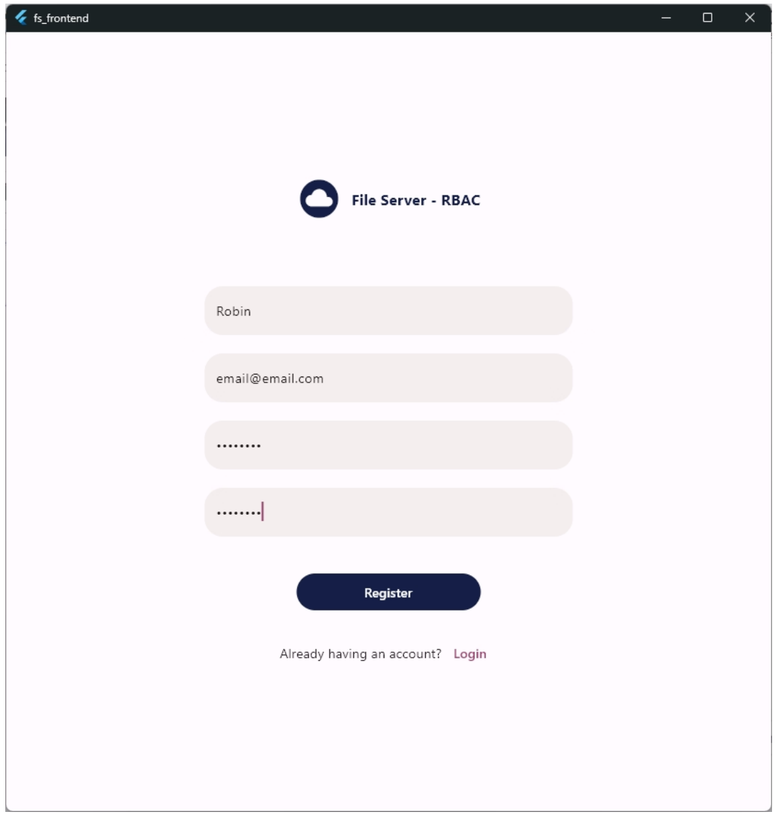
- Granular controls of roles and its usage
- Multiple workspaces for collective approach
- Enforcing security enforcements
- Secure JWT token authentication
- Optimized read and write speed using graph database (Neo4J)
- Security notifications with sign in emails

Frontend Features

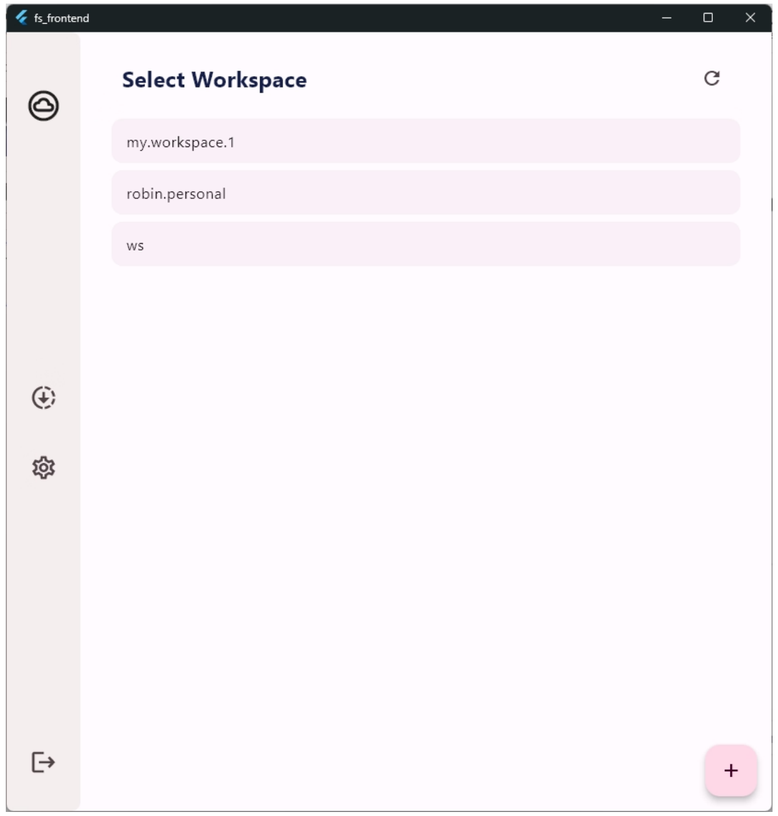
VIEW AND COMPONENTS



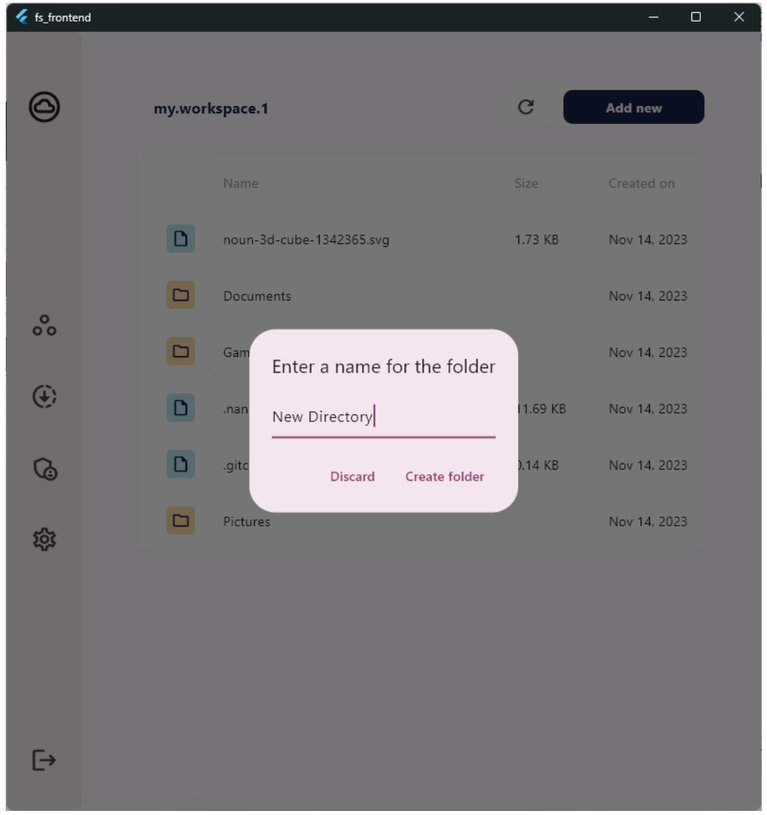
LOGIN SCREEN



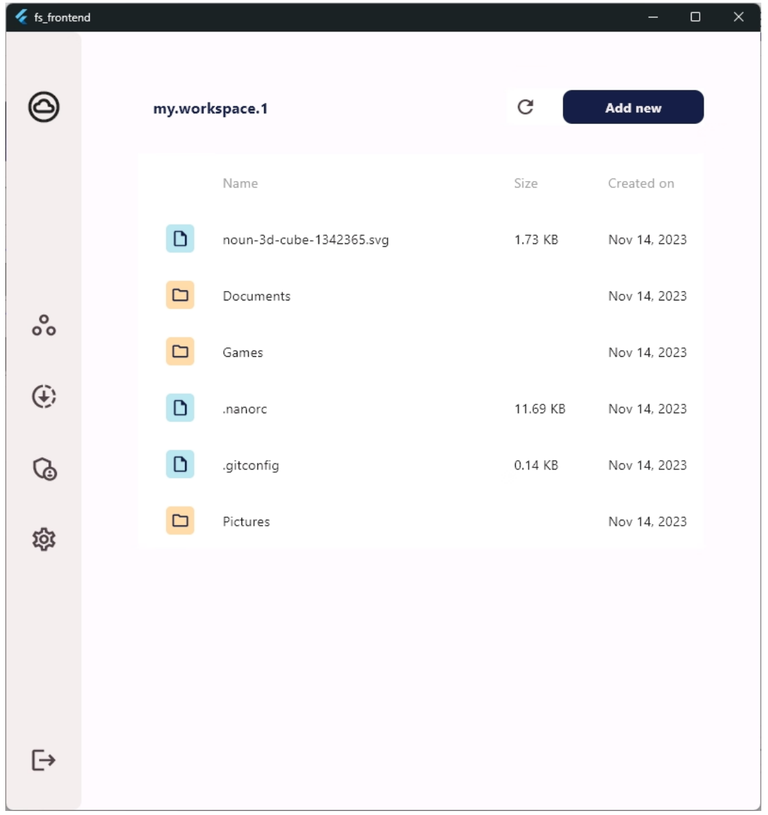
REGISTRATION SCREEN



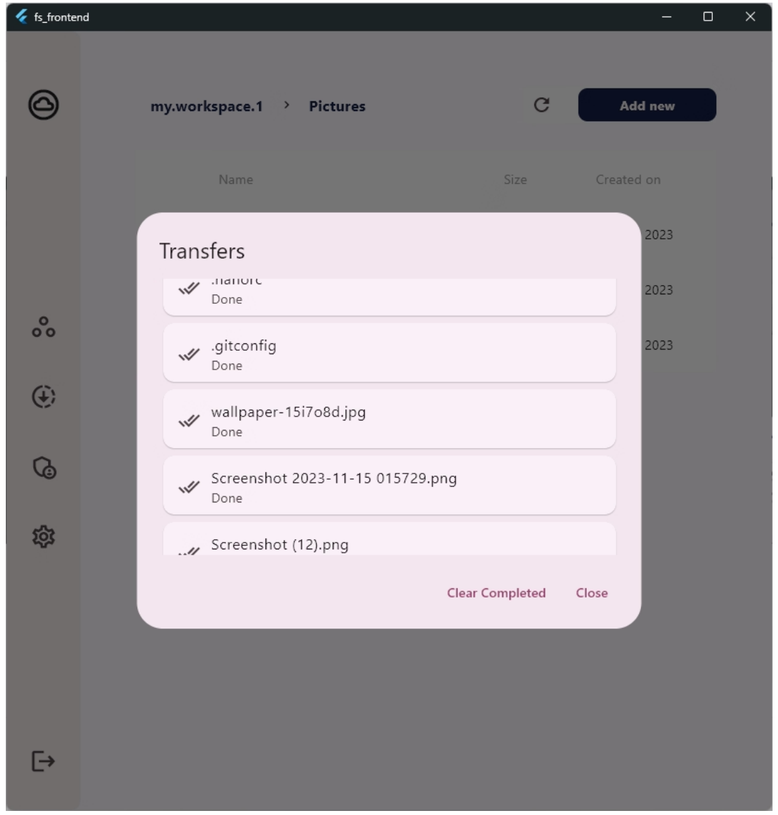
WORKSPACE SELECTION



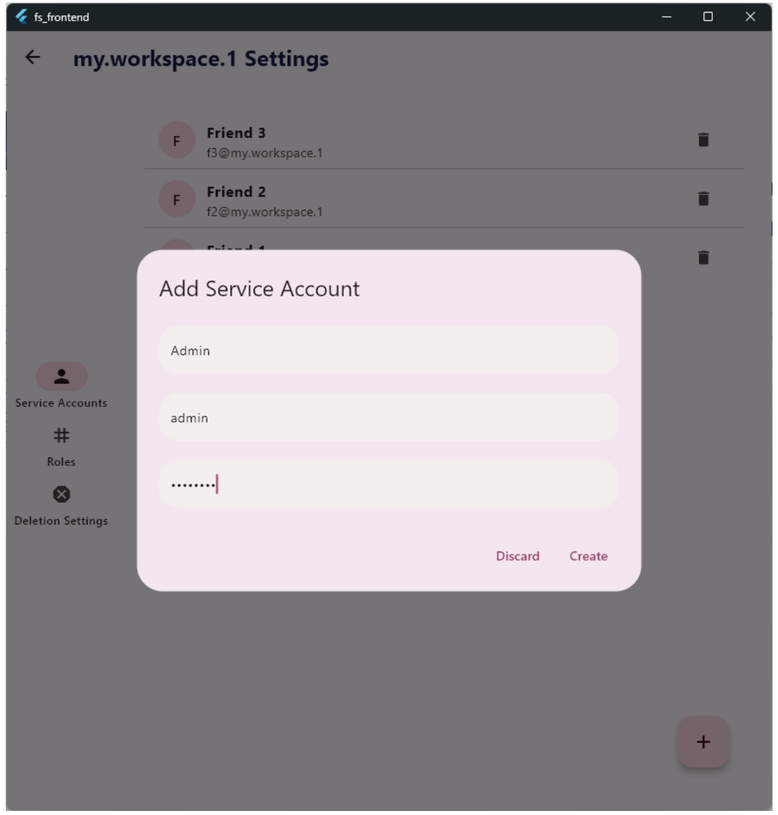
NEW FOLDER



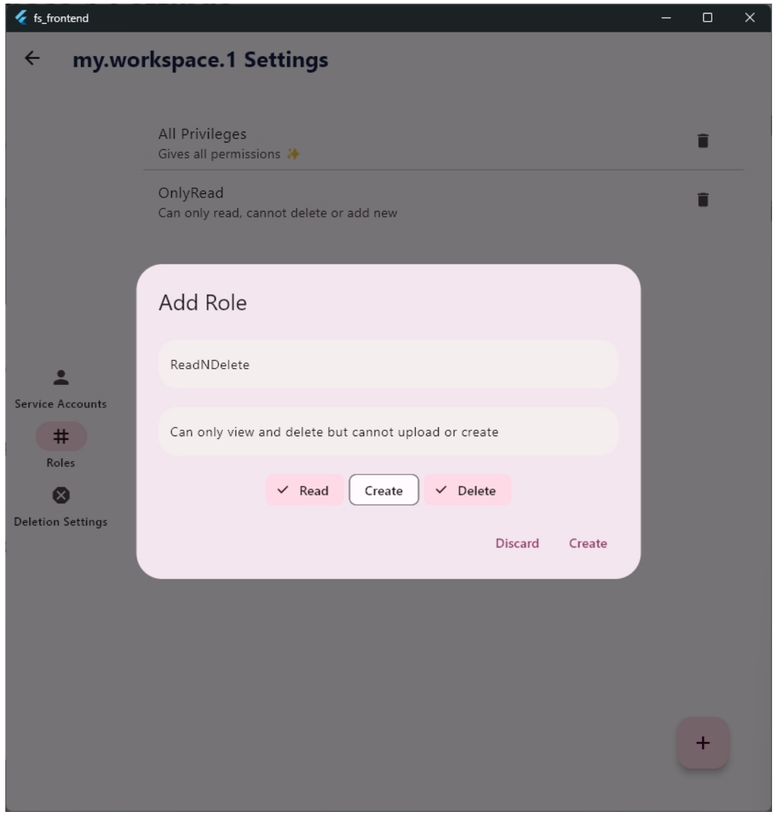
EXPLORER VIEW



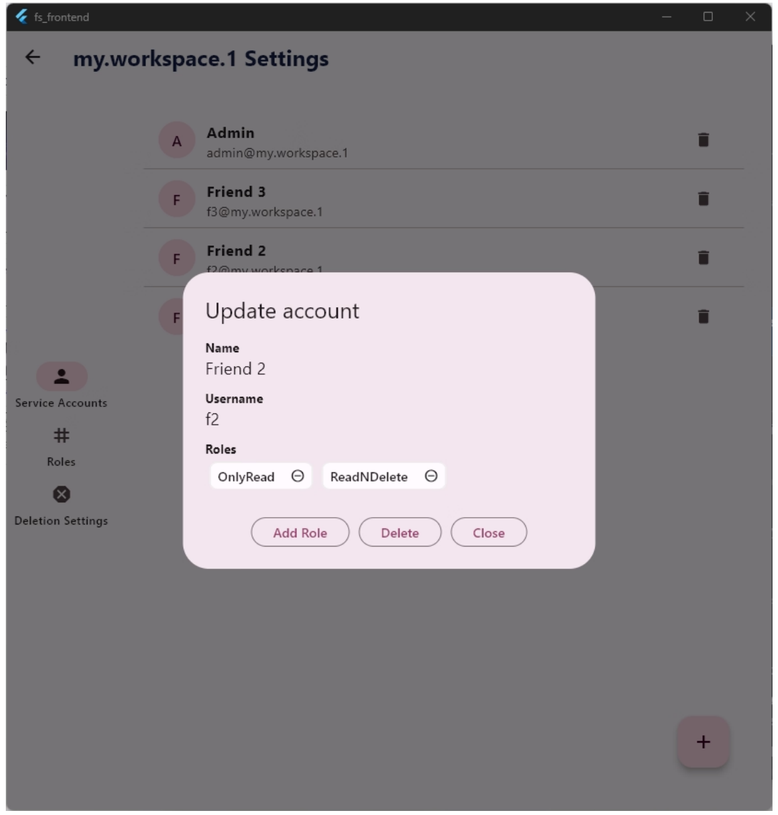
TRANSFERS



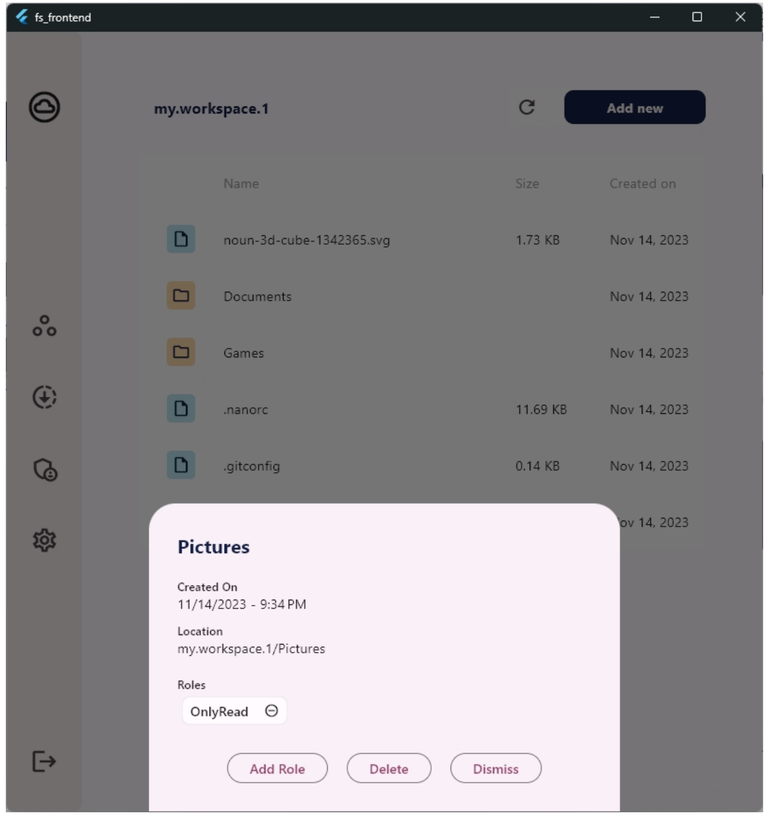
NEW SERVICE ACCOUNT



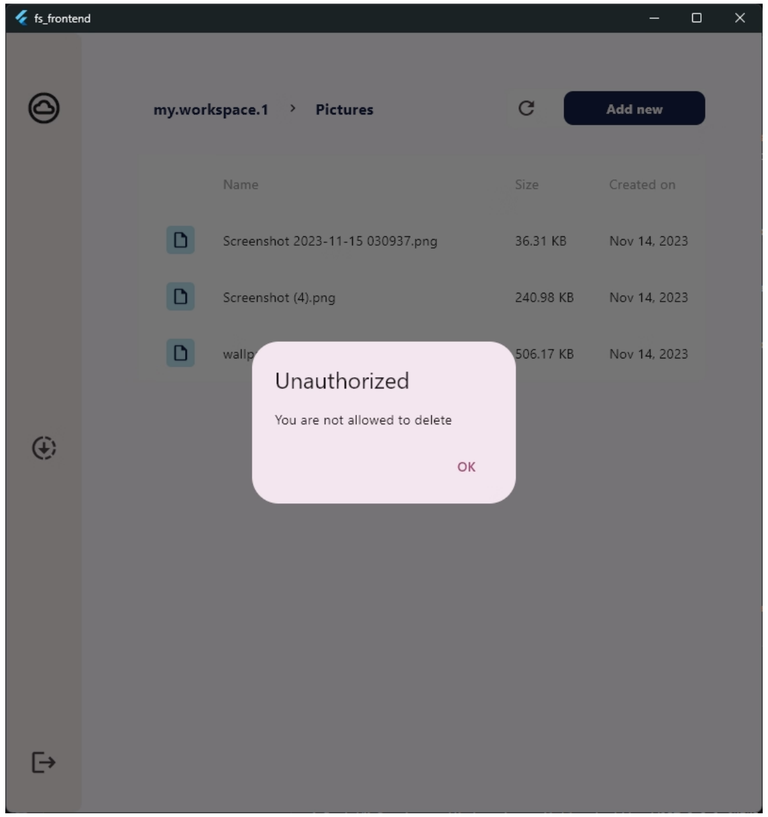
NEW ROLE



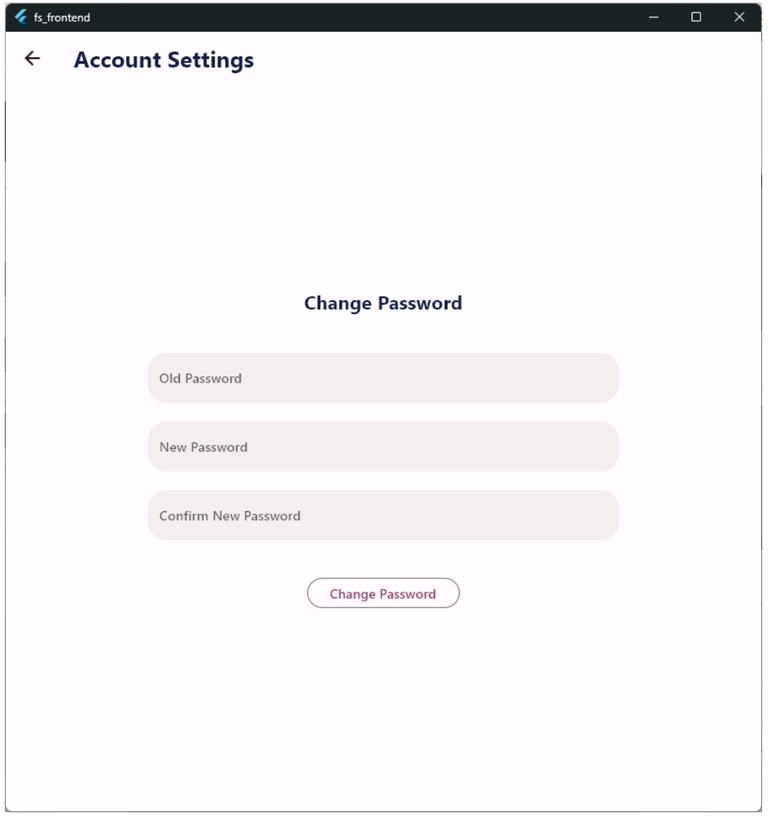
ACCOUNT WITH ROLES



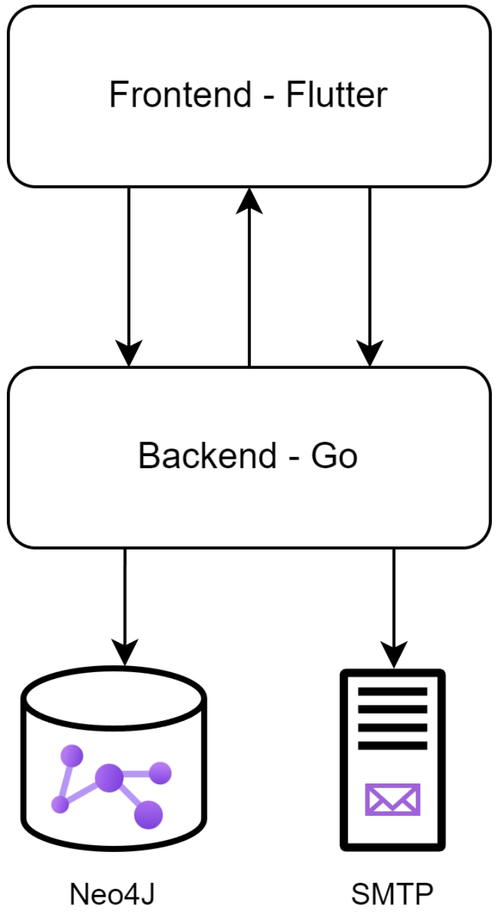
FOLDER DETAILS



PERMISSION DENIAL (RBAC)



PASSWORD RESET



- Frontend is created using Flutter.
  - With flutter, libraries like BLoC, File Picker with a lot more is also used.
- Backend is created using Go.
- For database requirements, Neo4J is used.
  - Neo4J is used for performance and query requirements.
  - Data that is stored is mostly in nested structure
- A SMTP server is also used for mail requirements. For testing and mocking purposes, MailHog is used.

Transfers & Communication

01

DB < - > SERVER < - > APP

The app communicates with the server with the help of HTTP route path with appropriate parameters, body and headers. The server directly connects to the Database and other related services to perform other actions.

02

UPLOAD AND DOWNLOAD

File transfers can range from file of size KBs to GBs. In order to make sure the upload is seamless. the upload and download happens by transferring data in chunks. This approach helps in make application memory efficient.

Security Checks

01

PASSWORDS

Each and every password that is used is made to follow strict requirements which include: having minimum of 8 characters with numbers and symbols. This password is then hashed to store in the database securely, making it a secure.

02

TOKENS

The authentication of users is done using JWT tokens. It includes various information about the user login. Details like created time, expiry time, source IP, user details and much more, help in validating the user’s future requests.

Graph Database

- Each entity is represented by nodes in a Graph Database.
- The file system and directory hierarchy is easy to be maintained and managed in a graph structure.

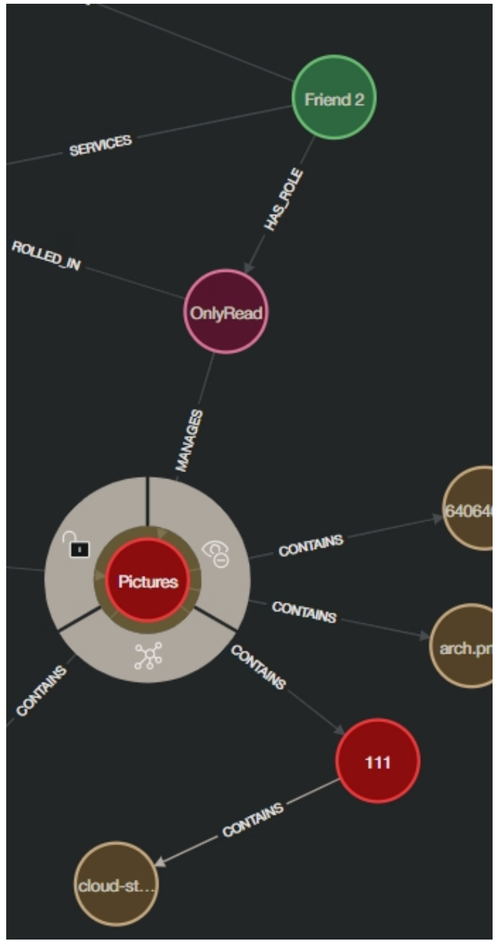
Types of Nodes

- Workspace
- OwnerAccount
- ServiceAccount
- Directory
- RootDirectory
- File
- ServiceAccount
- Role

Types of Relationships

- OWNS
- STORES
- CONTAINS
- SERVICES
- ROLLED_IN
- HAS_ROLE
- MANAGES

ROLE (SHORTEST PATH)



- The shortest path towards the root that leads to a assigned role is designed as the role to the specified resource.
- For instance (ref. picture)
  - The folder picture is assigned to the role `OnlyRead`.
  - Any item that belongs under the Pictures will resolve to the role `OnlyRead`.

MULTIPLE ROLE RESOLUTION

ResolutionMap	true	false
true	true	true
false	true	false

```
var resolutionMap map[bool]map[bool]bool = map[bool]map[bool]bool{
  true: {
    true:  true,
    false: true,
  },
  false: {
    true:  true,
    false: false,
  },
}
```

- Roles being lenient compared to default permissions.
- This is because roles are the only way a user gets permissions.
- If anyone of the roles gives permissions to an action, it is carried over to resolved permission.
- Implemented using a 2x2 Boolean map.