**Login Authentication and Management**

**Overview of the Application**

This application is a user authentication and management system built using Golang, SQLite, and JWT-based authentication. It allows users to register, log in, reset passwords, and provides an admin panel for managing users. Additionally, it implements Single Sign-On (SSO) with JWT, allowing users to authenticate once and access multiple parts of the system without logging in again.

**Why We Chose SSO with JWT**

Single Sign-On (SSO) is an authentication method that enables users to log in once and access multiple applications without needing to re-enter credentials. It is widely used for:

* Enhancing user experience (reducing repeated logins)
* Improving security (centralized authentication)
* Simplifying user management (fewer credentials to manage)

We chose JSON Web Token (JWT) for SSO because of its stateless nature, meaning:

* Tokens are self-contained, so the server does not need to store session data.
* Tokens can be securely signed and verified without database lookups.
* JWTs are scalable and work well in distributed systems.

Since our application is self-contained and does not need third-party authentication, we chose JWT-based SSO instead of OAuth.

**Steps to deploy and run the application:**

Run the application using the following command

**C:\Users\Dell\Pictures\Screenshots\Screenshot 2025-03-09 224042.png**

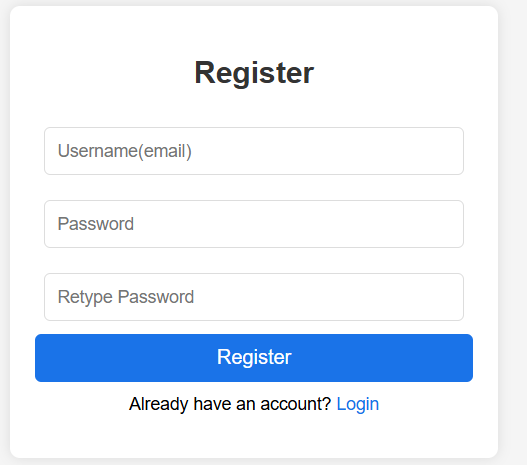
This will launch the golang server on port 8080

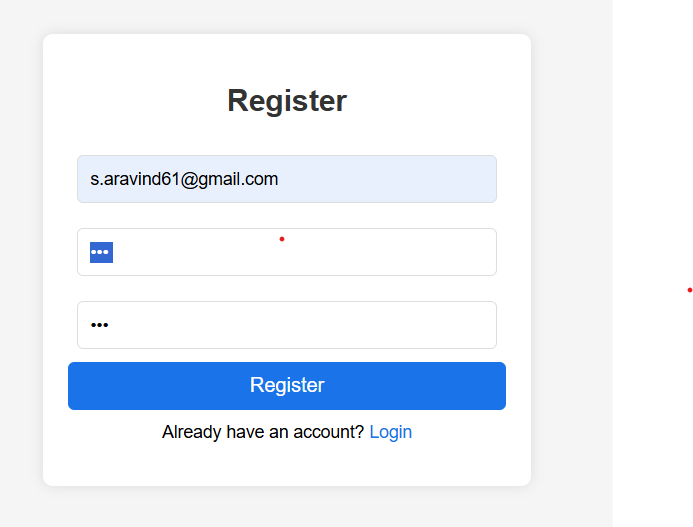
**User Registration**

Users register with a username and password.

Passwords are hashed using bcrypt library for password security and the hashed password is stored in database.

URL - <http://localhost:8080/register>





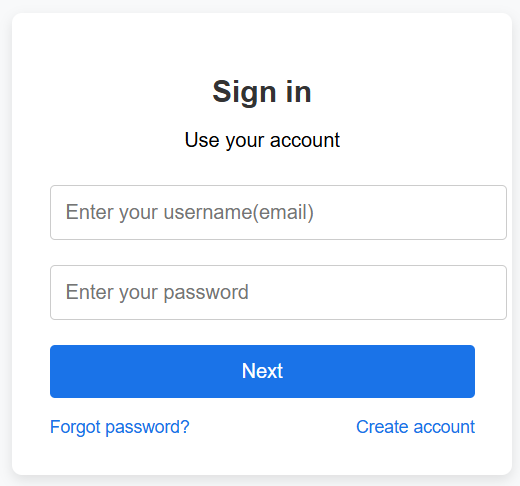
**User Login**

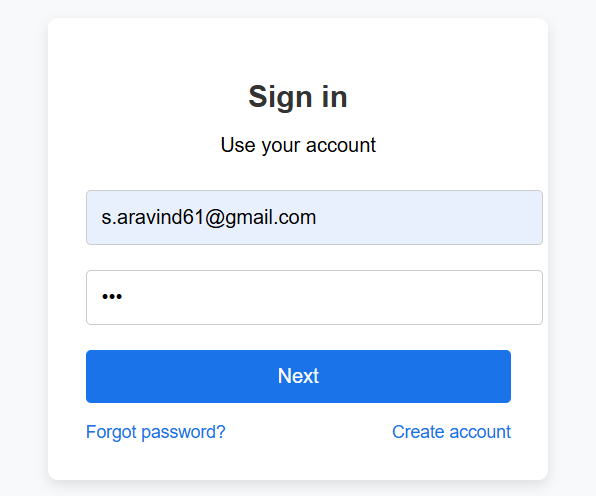
User logs in using the username and password

The system validates the credentials and generates JWT and stores them in a cookie

Upon successful JWT token validation user will be logged in successfully

URL - <http://localhost:8080/>

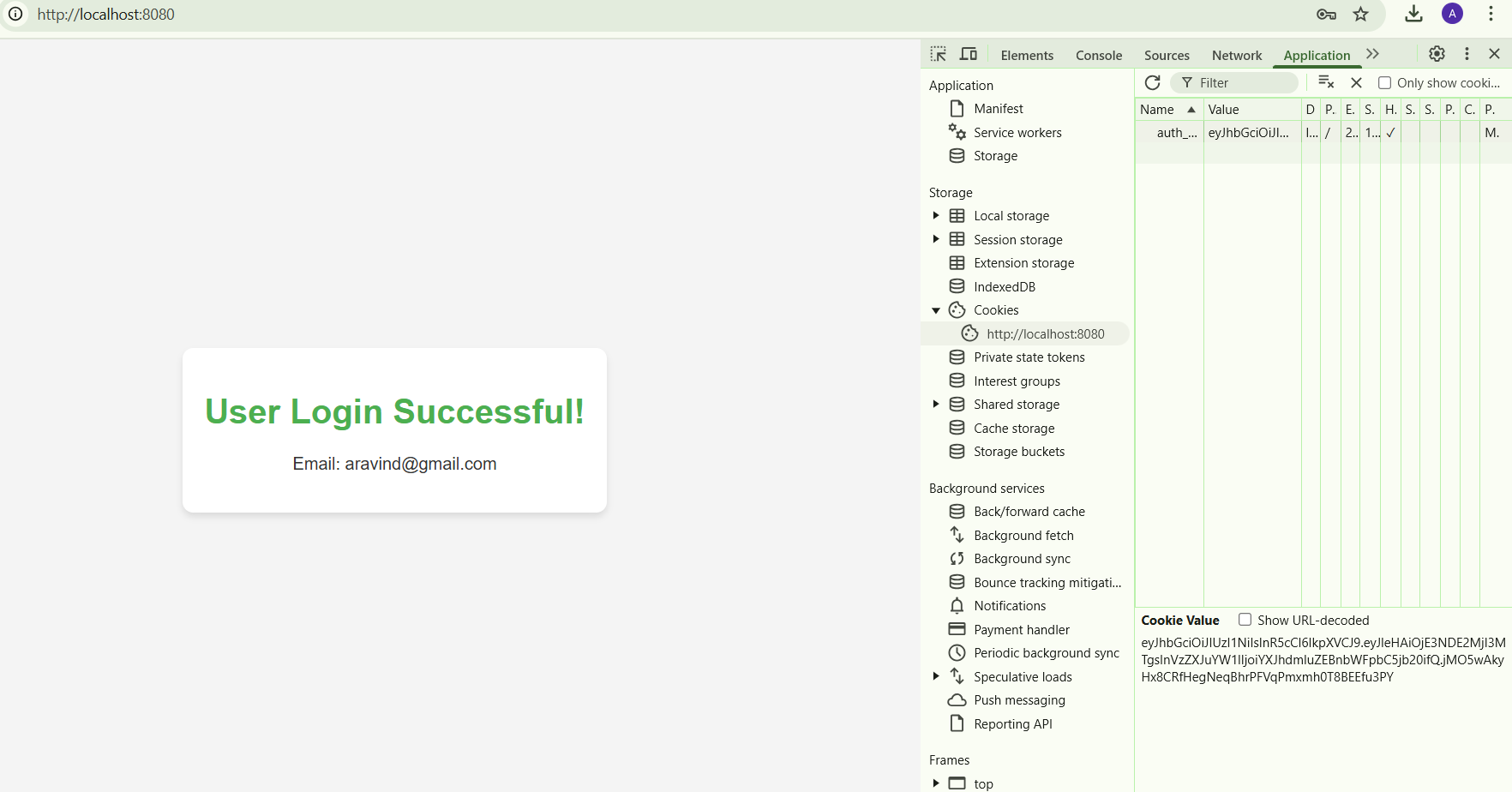




**JWT Token:**

Once user logs in , JWT token is generated and stored in cookie

The JWT token is read from cookie and used for further user authentication

****

**Admin flow:**

Admin can log into view admin dashboard using the following credentials

Username – admin

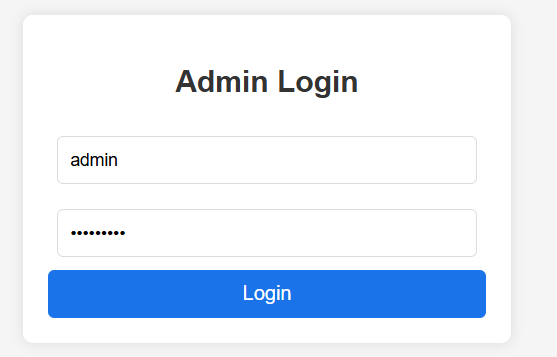
Password – admin@123

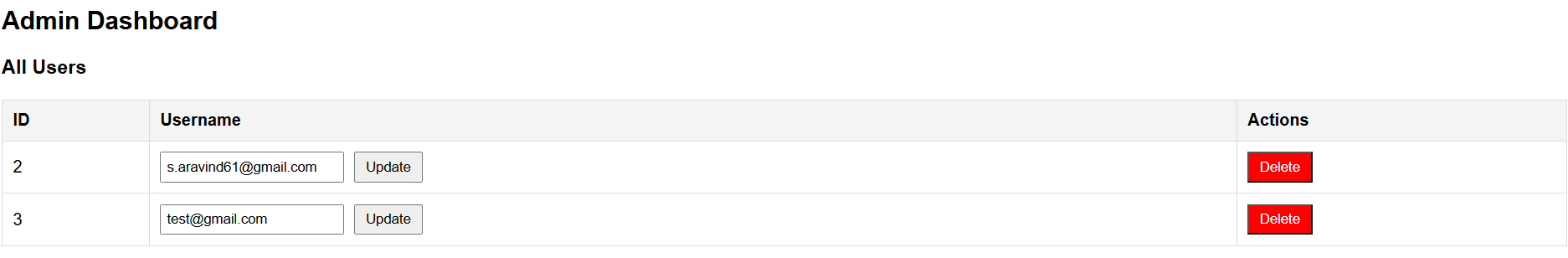
The system validates the credentials and generates JWT and stores them in a cookie

Upon successful JWT token validation admin will be able to view admin dashboard

Admin dashboard displays all user information and option to update and delete users

URL - <http://localhost:8080/admin>

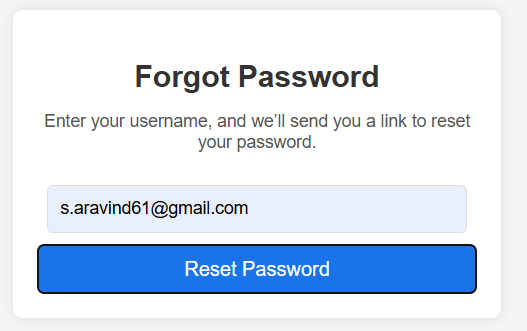
****

****

**Forgot/Reset password:**

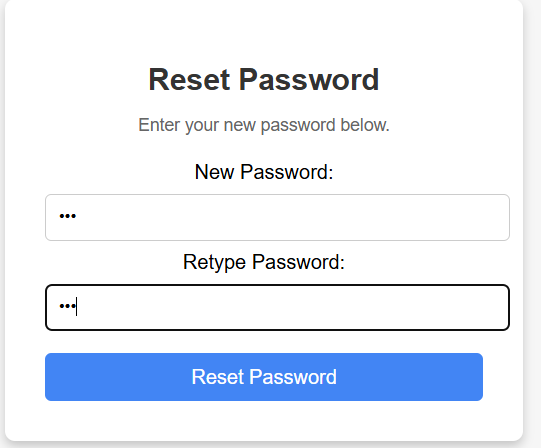
Once user enters email in forgot password , a link will be generated with token and sent to user email.

URL - <http://localhost:8080/forgot-password>



Once user clicks on reset password link from email, token will be validated and user can updated the password

Sample reset password link - <http://localhost:8080/reset-password?token=MrM08ZIOX2Nhv7mXIvzB> ‘

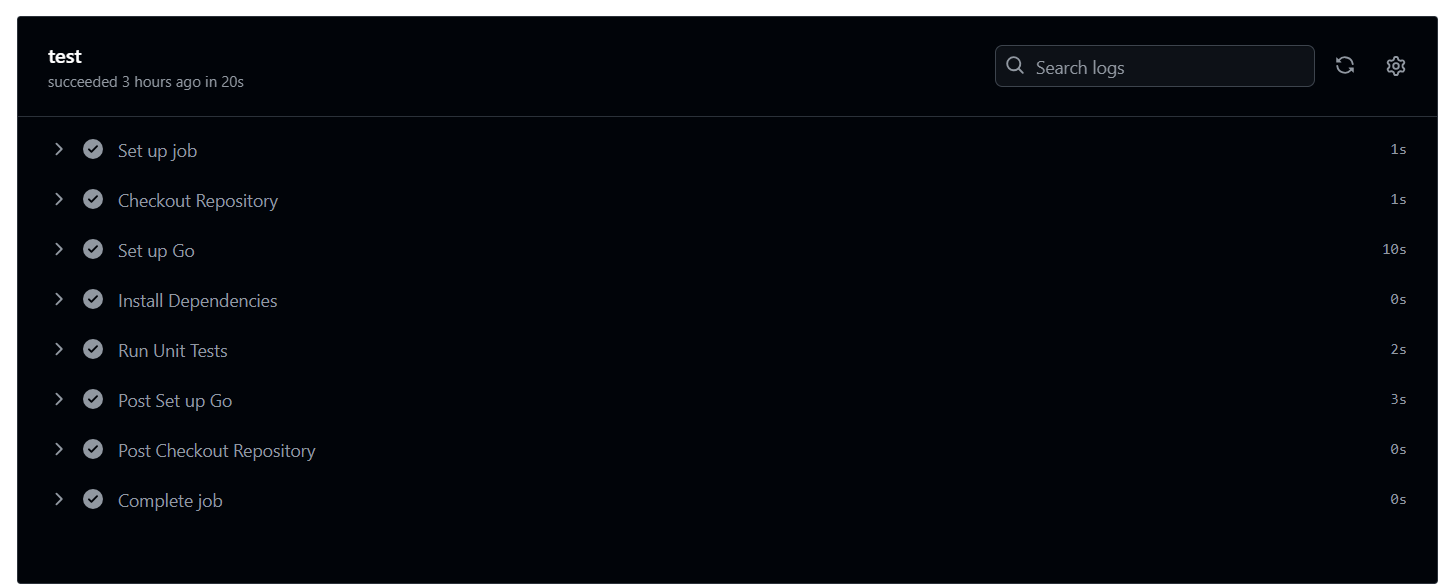


**Unit test pipeline:**

This pipeline installs dependencies and runs unit tests on every commit.

If all unit tests pass, pipeline succeeds.

Pipeline link - <https://github.com/aravindsiva61/jwt_login_management/actions/runs/13739366944/job/38426800490>



**Release pipeline:**

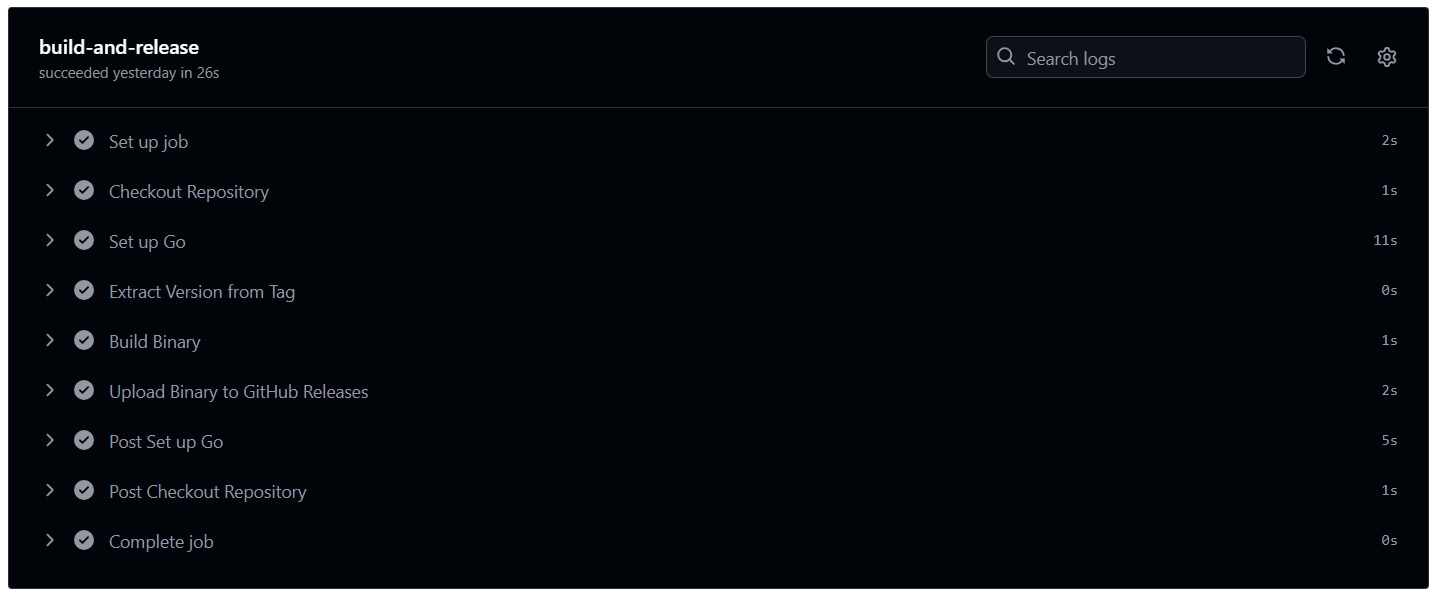
This pipeline runs on creating a new release in github.

Build version maps to the tag created in release.

The binary is uploaded to github release folder

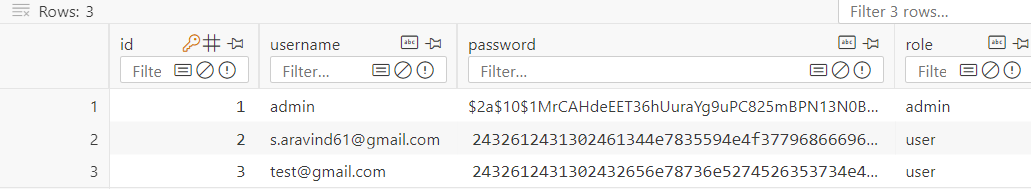
Binary link - <https://github.com/aravindsiva61/jwt_login_management/releases/tag/v1.0.0>

Pipeline link - <https://github.com/aravindsiva61/jwt_login_management/actions/runs/13739384861/job/38426835158>



**SQLite database:**

We used SQLite database to store user credentials in users table and password reset token in password reset table



**Unit testing:**

Unit tests are written using golang unit testing framework and covers positive and negative flows.

Unit testing files – **handlers\_test.go, db\_test.go**

