Note the following for:

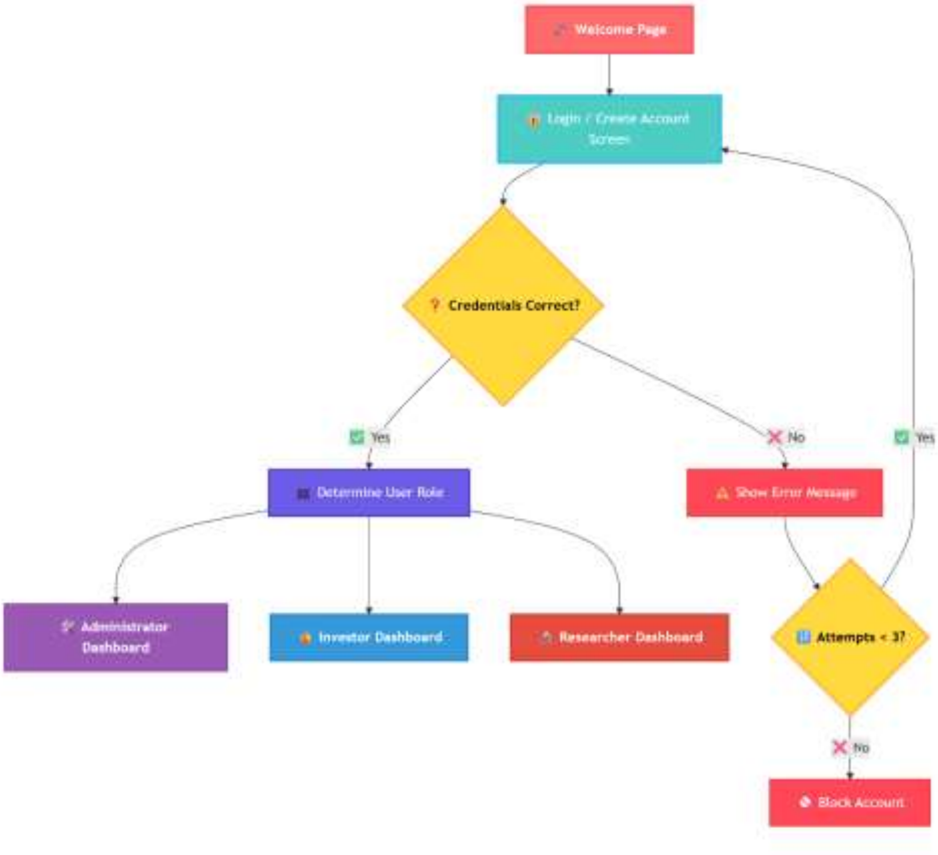
Admin Data:
When you run app.py first time, the app will auto-create a default admin:

- **Email: admin@example.com**
- **Password: Admin@1234**
- **Username: system.admin.sou.min**

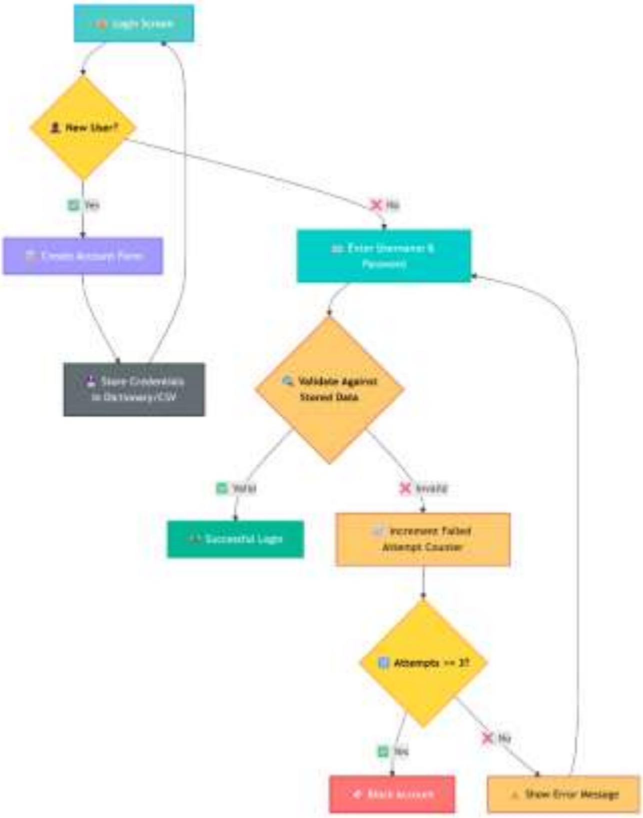
Pseudo Code and Flow Charts of the main functioning of the app:

1. Alogrithm Sytem Overview Diagrams:

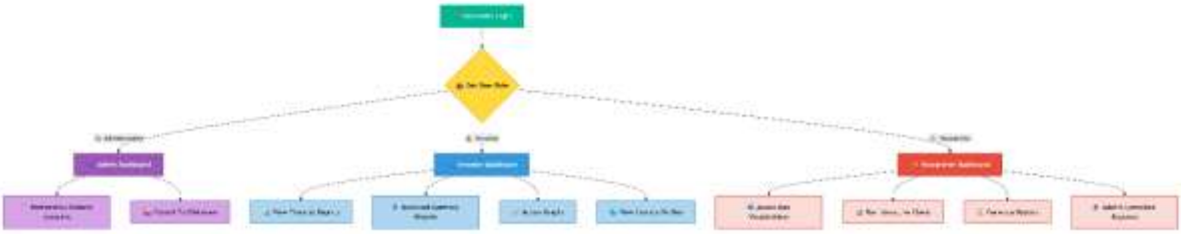
1.1 App flow



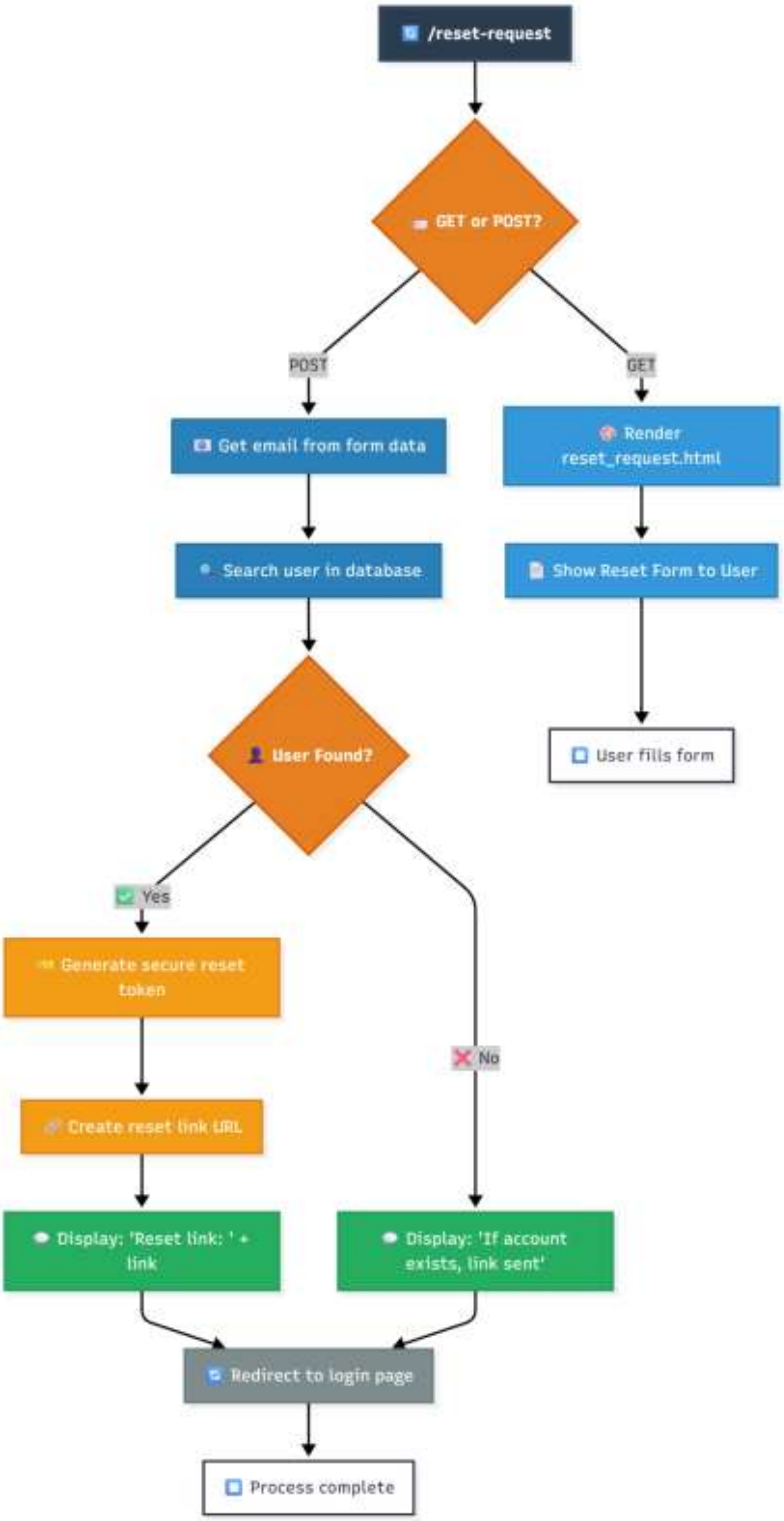
1.2 User Login Flow



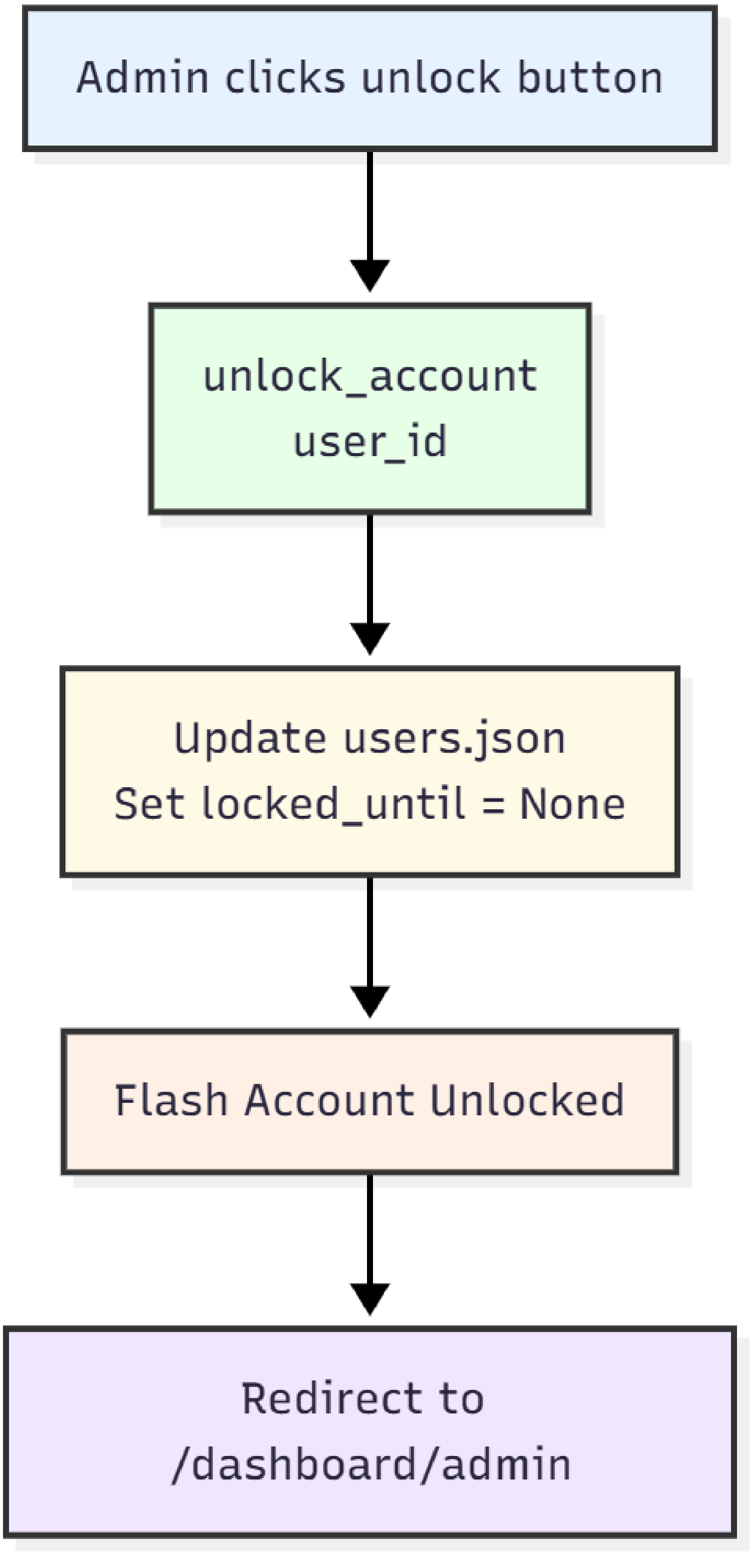
1.3 Dashboard Redirection Flow



1.4 Password Reset Flow

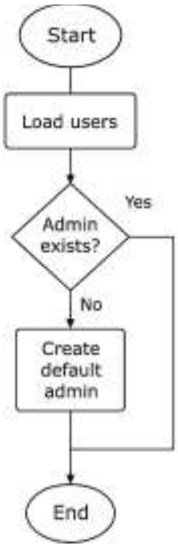


1.5 Administrator Unlock Flow



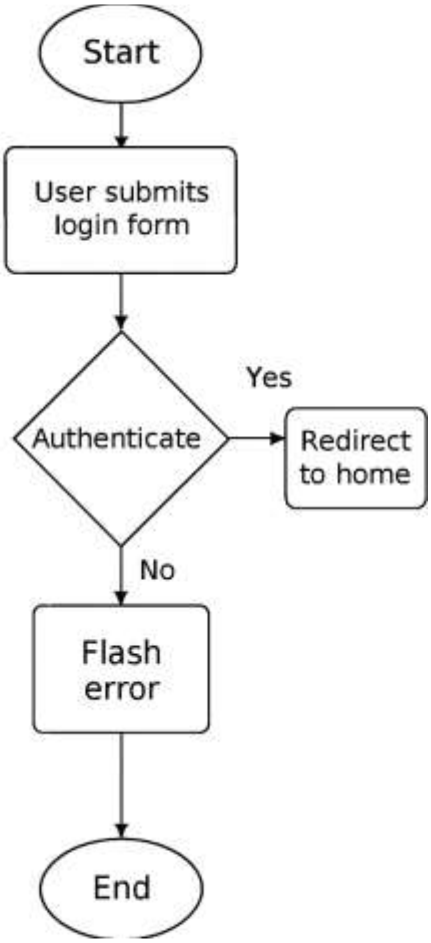
2. PSEUDOCODE AND FLOWCHARTS FOR ALL FUNCTIONS:

2.1 create_initial_admin()



```
FUNCTION create_initial_admin():
  users_list = load_users().get("users", [])
  IF no user with role == "Administrator" THEN
    admin = create_user(
      first="System",
      last="Admin",
      email="admin@example.com",
      country="SouthAfrica",
      organization="MINN",
      role="Administrator",
      password="Admin@1234"
    )
    PRINT "Default admin created"
  ENDIF
END FUNCTION
```

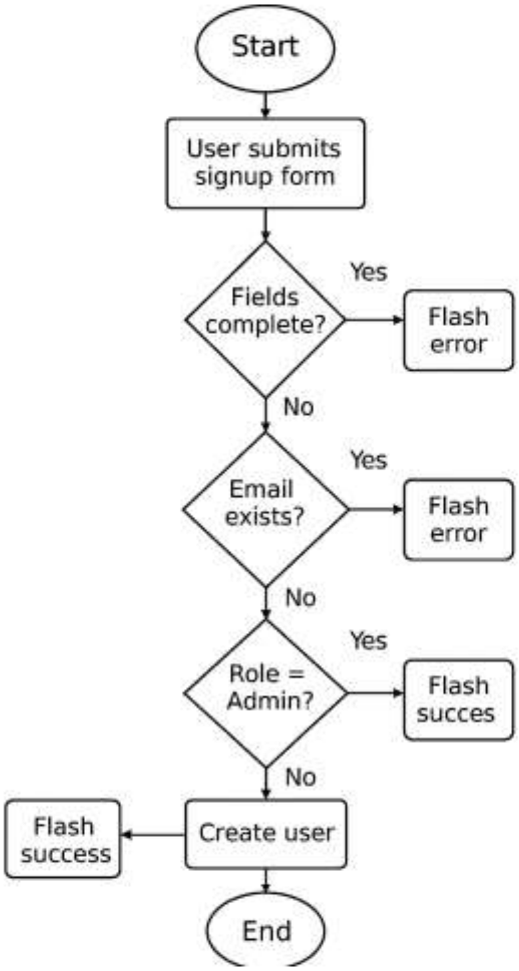
2.2 login()



```
ROUTE /login (methods = GET, POST):


IF request.method == POST THEN
  username = form.username
  password = form.password
  success, message = authenticate(username, password)


  IF success THEN
    flash("Login successful")
    redirect to home
  ELSE
```

```
    flash(message)
  ENDIF
ELSE
  render login.html
ENDIF
```

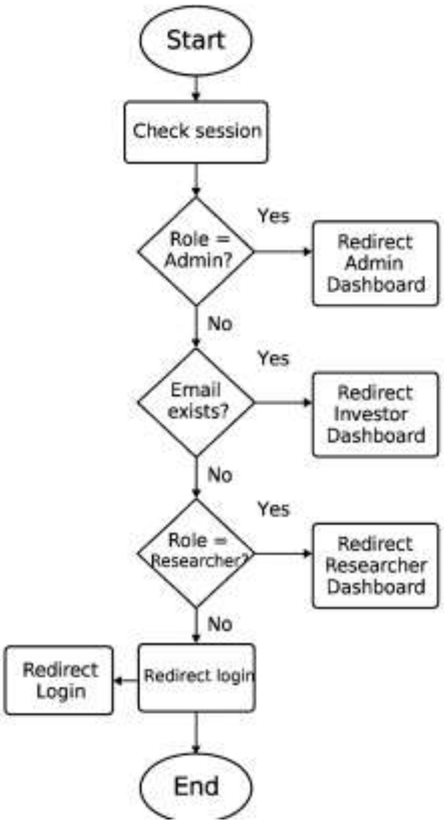
2.3 signup()



```
ROUTE /signup (methods = GET, POST):


IF request.method == POST THEN
  Collect form inputs
  IF any field missing THEN
    flash("Missing fields")
```

```
        return signup.html
    ENDIF

    IF password != confirm THEN
        flash("Passwords do not match")
        return signup.html
    ENDIF

    IF find_user_by_email(email) THEN
        flash("Email already exists")
        return signup.html
    ENDIF

    IF role == "Administrator" THEN
        flash("Admins cannot self-register")
        return signup.html
    ENDIF

    new_user = create_user(first, last, email, country, org, role, password)
    flash("Account created.  Please login.")
    redirect to /login
ELSE
    render signup.html
ENDIF
```

2.4 dashboard_admin()



```
ROUTE /dashboard/admin:


REQUIRE ROLE = "Administrator"

users = load_users()

role_counts = {"Administrator":0, "Investor":0, "Researcher":0}

FOR each user IN users:

  role_counts[user.role] += 1

ENDFOR


render dashboard_admin.html(users, role_counts)
```

2.5 dashboard_investor()

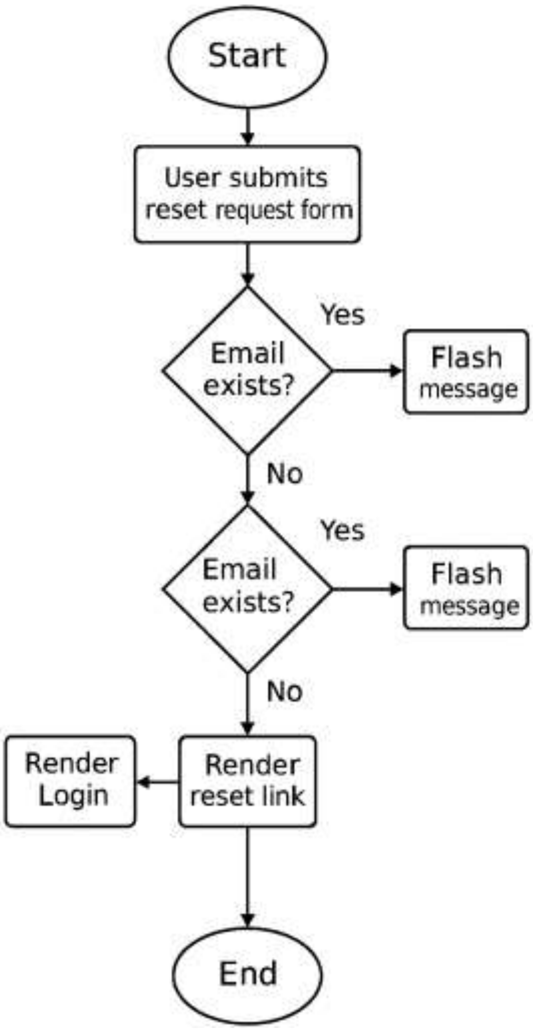
```
ROUTE /dashboard/investor:


REQUIRE ROLE = "Investor"
```

```
minerals = load_minerals_json()
chart_html = generate_overview_chart()
map_path = generate_africa_mineral_map(minerals)
render dashboard_investor.html(chart_html, map_path)
```

2.6 dashboard_researcher()

```
ROUTE /dashboard/researcher:

REQUIRE ROLE = "Researcher"
minerals = load_minerals_json()
map_path = generate_africa_mineral_map(minerals)
render dashboard_researcher.html(minerals, map_path)
```

2.7 reset_request()



```
ROUTE /reset-request (GET, POST):


IF POST:
  email = form.email
  user = find_user_by_email(email)
  IF user EXISTS:
    token = generate_reset_token(email)
    reset_link = url_for('reset_password_route', token)
    flash("Link generated: " + reset_link)
  ELSE:
    flash("If account exists, link sent.")
  redirect to login
ELSE:
  render reset_request.html
```

2.8 reset_password_route(token)

```
ROUTE /reset/<token> (GET, POST):

email = verify_reset_token(token)
IF NOT email:
  flash("Invalid token")
  redirect to /reset-request

IF POST:
  pw = form.password
  confirm = form.confirm
  IF pw != confirm:
    flash("Passwords do not match")
    return reset_password.html
  reset_password(email, pw)
  flash("Password reset. Please login.")
  redirect to login
ELSE:
  render reset_password.html
```

2.9 admin_unlock(user_id)

```
ROUTE /admin/unlock/<user_id> (POST):

REQUIRE ROLE = "Administrator"
ok = unlock_account(user_id)
IF ok:
  flash("Account unlocked.")
ELSE:
  flash("User not found.")
redirect to /dashboard/admin
```

2.10 minerals_dashboard()

```
ROUTE /minerals:


minerals = load_minerals_json()
chart_html = generate_overview_chart()
render minerals_dashboard.html(minerals, chart_html)
```

2.11 mineral_detail_chart(mineral_name)

```
ROUTE /minerals/<mineral_name>:


chart_html = generate_mineral_chart(mineral_name)
render minerals_dashboard.html(chart_html, single=True)
```