# Project Plan: SecureCalc – A Secure Calculator Web App

## Objective

Develop a secure web application where users can:  
- Register and log in (credentials stored in a file)  
- Perform any of six arithmetic operations  
- View their results  
- Log out or navigate back to the home page

## Key Features

- User registration and login  
- File-based credential storage (e.g., CSV or JSON)  
- Arithmetic functions: Add, Subtract, Multiply, Divide, Modulus, Power  
- Secure session handling  
- Logout and home navigation

## Project Structure

securecalc/  
│  
├── app.py # Main Flask application  
├── users.txt / users.json # File-based user storage  
├── templates/  
│ ├── base.html  
│ ├── login.html  
│ ├── signup.html  
│ ├── home.html  
│ └── result.html  
├── static/  
│ └── style.css # Optional CSS styling  
└── requirements.txt

## Technologies Used

|  |  |
| --- | --- |
| Component | Tool/Library |
| Backend | Flask |
| Frontend | HTML, CSS |
| Storage | File (CSV/JSON) |
| Security | Werkzeug Password Hashing, Flask sessions |

## Development Timeline (3 Weeks)

### Week 1: Setup & User Management

- Initialize Flask app and structure  
- Create signup and login pages  
- Hash and store passwords in users.txt or users.json  
- Implement session-based authentication

Deliverables:  
- Secure registration/login system  
- File-based credential storage

### Week 2: Calculator Core

- Create home.html with 6 operation buttons  
- Implement arithmetic logic in backend  
- Pass and display results in result.html  
- Add error handling (e.g., divide by zero)

Deliverables:  
- Working calculator  
- User-specific session with result display

### Week 3: Finishing Touches

- Implement logout route  
- Add navigation controls (back to home, logout)  
- Style with basic CSS/Bootstrap  
- Test all routes and operations  
- Finalize documentation (README.md)

Deliverables:  
- Clean, user-friendly interface  
- Fully functional, testable app

## Security Considerations

- Use werkzeug.security for password hashing (generate\_password\_hash, check\_password\_hash)  
- Do not store plain-text passwords  
- Use Flask session securely (with a secret\_key)  
- Validate and sanitize user inputs

## Optional Enhancements

- Store user history in the file  
- Add session timeout  
- Input validation for numeric-only inputs