

GoWorkout Project Documentation

1. Project Overview

GoWorkout is a web-based fitness platform designed to help users achieve their health and training goals. The application provides customizable workout plans, a daily calorie calculator, and user profile management. It features a client-side authentication system and persists user data locally using browser storage, making it a lightweight and privacy-focused tool for fitness tracking and workout planning.

Key Objectives

- Provide users with structured, progressive workout programs
- Calculate personalized daily calorie intake recommendations
- Enable secure user account management with session persistence
- Deliver an intuitive interface for fitness planning and tracking

2. Core Features

2.1 User Authentication System

The application implements a comprehensive authentication system with the following capabilities:

Sign Up

- Users create accounts by providing:
 - First Name
 - Second Name
 - Username (must be unique)
 - Password (minimum 6 characters)
- Validation checks ensure all fields are completed and passwords meet security requirements
Username uniqueness is verified before account creation
-

Log In

- Secure credential verification against stored user database
- Session management through `gw_currentUser` local storage key
- Failed login attempts display appropriate error messages

Session Management

- Users remain logged in across page navigations
- Session persists until manual logout or browser cache clearance
- Protected routes redirect unauthenticated users to login page

Access Control

- Training Programs page requires authentication
- User Profile page is restricted to logged-in users
- Automatic redirects ensure security of protected resources

2.2 Training Programs

Three progressive difficulty levels tailored to different fitness experience levels:

Beginner Program

Focus on establishing proper form, building confidence, and creating a sustainable routine.

- Day 1: Full Body Basics
 - Bodyweight Squats: 3 sets × 8 reps
 - Push-ups: 3 sets × 8 reps
 - Walking Lunges: 3 sets × 8 reps
 - Plank: 3 sets × 10 seconds
- Day 2: Rest
- Day 3: Strength Foundations
 - Glute Bridges: 3 sets × 8 reps
 - Wall Push-ups: 3 sets × 8 reps
 - Bird Dogs: 3 sets × 8 reps
 - Superman Holds: 3 sets × 10 seconds

Intermediate Program

Balanced split routine introducing equipment and increased volume.

- Day 1: Upper Body — Push-ups, Dumbbell Rows, Shoulder Press, Tricep Dips (4×10)
- Day 2: Lower Body — Squats, Lunges, Calf Raises (4×10)
- Day 3: Cardio — Planks, Mountain Climbers, Bicycle Crunches (4×30s)
- Day 4: Rest

Advanced Program

High-intensity Push/Pull/Legs split for experienced athletes.

- Day 1: Push — Bench Press, Military Press, Dips (5×12)
- Day 2: Pull — Pull-ups, Barbell Rows, Face Pulls (5×12)
- Day 3: Legs — Squats, Deadlifts, Lunges (5×12)
- Day 4: Rest

2.3 Calorie Calculator

Personalized daily caloric intake recommendation based on individual metrics.

Input Parameters

- Weight (kg)
- Height (cm)
- Age (years)
- Gender (Male/Female)
- Goal (Lose Weight, Maintain, Gain Muscle)

Calculation Methodology

The calculator employs the **Maintain Jeor Equation** to determine Basal Metabolic Rate (BMR):

For males: $BMR = (10 \times W) + (6.25 \times H) - (5 \times A) + 5$

For females: $BMR = (10 \times W) + (6.25 \times H) - (5 \times A) - 161$

Where:

- W = Weight in kilograms
- H = Height in centimeters
- A = Age in years

Activity Multiplier

Daily caloric needs = $BMR \times 1.55$ (sedentary to light activity level)

Goal-Based Adjustment

- **Lose Weight:** Subtract 500 calories/day (approximately 0.5 kg/week deficit)
- **Maintain:** No adjustment applied
- **Gain Muscle:** Add 500 calories/day (approximately 0.5 kg/week surplus)

2.4 User Profile

Displays authenticated user information and account management options.

Profile Information Display

- First Name
- Second Name
- Username

Account Management

- Logout functionality to end active sessions
- Secure session termination clearing `gw_currentUser`

3. Technical Architecture

3.1 Technology Stack

Technology	Purpose
HTML5	Semantic markup and page structure
CSS3	Responsive styling and layout
JavaScript (ES6+)	Dynamic functionality and DOM manipulation
LocalStorage API	Client-side persistent data storage
Node.js	Testing environment and test execution

Table 1: Technology Stack Overview

3.2 Project File Structure

File	Purpose
index.html	Landing page and application entry point
login.html	User authentication login interface

signup.html	User registration and account creation form
trainingpro.html	Training programs display (authentication required)
caloriecalculator.html	Calorie calculation tool interface
profile.html	User profile display and account management
script.js	Core application logic and business functions
style.css	Global styling and responsive design rules
test.js	Unit test suite for core functions
README.md	Project introduction and overview

Table 2: Project File Structure

4. Code Architecture

4.1 Core Functions in script.js

Authentication Functions

- `signup(event)` — Form submission handler for new user registration
- `login(event)` — Credential verification and session initialization
- `logout()` — Session termination and data cleanup
- `isLoggedIn()` — Boolean check for active authentication
- `getCurrentUser()` — Retrieves current session username
- `setCurrentUser(username)` — Initializes new session

User Data Management

- `addUser(user)` — Adds new user to persistent storage
- `findUser(username)` — Locates user by username
- `getUsers()` — Retrieves all users from localStorage
- `saveUsers(users)` — Persists user array to localStorage

Feature Functions

- `calculateCalories()` — Computes daily caloric recommendations
- `showProgram(level)` — Renders workout program to DOM
- `initProfile()` — Initializes user profile page
- `goToPrograms()` — Navigation with authentication check
- `updateNav()` — Dynamic navigation menu rendering

4.2 Data Persistence Layer

The application uses browser LocalStorage for data persistence:

User Storage Schema

```
// gw_users: Array of user objects
```

```
{ firstName: string,  
secondName: string,  
username: string,  
password: string }
```

```
// gw_currentUser: String (username)
```

Data Flow

- User registration → addUser() → saveUsers() → LocalStorage
- User login → loginUser() → Session validation → setCurrentUser() → LocalStorage
- Session check → getCurrentUser() → LocalStorage retrieval

4.3 Input Validation

The application implements robust validation throughout:

Authentication Validation

- All signup and login fields are required
- Password minimum length: 6 characters
- Username uniqueness checked before account creation
- Duplicate username prevention with user feedback

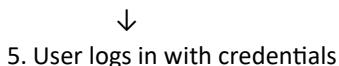
Calorie Calculator Validation

- All input fields must be populated
- Weight, height, and age must be positive numbers
- Invalid number format detection
- Empty field detection with user alerts

5. User Workflows

5.1 New User Registration

1. User accesses signup.html
↓
2. User enters First Name, Second Name, Username, Password
↓
3. Form submission triggers signup(event)
↓
4. Validation checks:
 - All fields populated?
 - Password ≥ 6 characters?
 - Username doesn't already exist?
↓
- 5a. Validation passes: User saved, redirect to login.html
- 5b. Validation fails: Alert displayed, form remains on page



5.2 Existing User Login

1. User accesses login.html
↓
2. User enters Username, Password
↓
3. Form submission triggers login(event)
↓
4. Credential verification:
 - Username exists?
 - Password matches?
↓
5a. Login successful: Session created, redirect to trainingpro.html
5b. Login failed: Alert displayed, form remains on page

5.3 Using the Calorie Calculator

1. User accesses caloriecalculator.html
↓
2. User enters Weight, Height, Age, Gender, Goal
↓
3. User clicks Calculate button
↓
4. calculateCalories() validation: All fields populated?
 - Numbers are positive?
 - ↓
5a. Validation passes: BMR calculated, daily target displayed
5b. Validation fails: Alert explains missing or invalid input
↓
5. Result shows personalized calorie recommendation

6. Installation & Setup

6.1 System Requirements

- Modern web browser (Chrome 90+, Firefox 88+, Edge 90+, Safari 14+)
- JavaScript enabled in browser settings
- LocalStorage access enabled (standard in all browsers)

6.2 Installation Steps

1. **Download** all project files to a local directory
2. **Preserve** directory structure with all HTML, CSS, and JS files together
3. **Open** index.html in your web browser
4. **Allow** any browser prompts for local storage access
5. **Navigate** using application menu links **6.3 First-Time Setup**

- Create a new account via signup page
- Set your password (minimum 6 characters)
- Log in with your credentials
- Explore training programs and use the calorie calculator

7. Testing

7.1 Test Suite Overview

The test.js file contains automated unit tests validating core functionality. Tests mock browser objects to enable Node.js execution.

Test Coverage

- testShowProgram() — Validates program rendering for beginner and advanced levels
- testCalculateCalories() — Tests calorie calculation with various scenarios
- Error handling validation for missing inputs
- Gender-specific BMR calculation verification
- Goal-based calorie adjustment verification

7.2 Running Tests

Prerequisites: Node.js installed on system

Execution:

```
cd [project-directory] node
test.js
```

Expected Output: Console logs showing test results for each scenario, including calculated values and program details.

7.3 Test Scenarios

The test suite validates:

- Beginner program HTML generation
- Advanced program HTML generation
- Male calorie calculation with maintenance goal
- Female calorie calculation with weight loss goal
- Error handling for missing required fields

8. Best Practices & Guidelines

8.1 Security Considerations

- **Passwords:** Currently stored in plaintext LocalStorage (not suitable for production). In production, implement backend authentication with encrypted password storage. **Session**
- **Management:** LocalStorage is accessible to client-side JavaScript. Use HttpOnly cookies for production applications.
- **Input Sanitization:** Implement additional HTML escaping for production deployment.

8.2 Maintenance & Extension

Adding New Workout Programs

- Update the programs object in script.js with new difficulty levels
- Follow existing naming convention and array structure
- Add corresponding button/link in trainingpro.html

Expanding Calorie Calculation

- Modify activity multiplier (currently 1.55) based on user activity level selection
- Implement additional calculation methods (Katch-McArdle, Harris-Benedict)
- Add maintenance dose tracking and history

User Dashboard Enhancement

- Add workout progress tracking
- Implement calorie logging and history
- Create personalized recommendations based on user data **8.3**

Browser Compatibility

- **LocalStorage:** Supported in all modern browsers and IE8+
- **ES6 JavaScript:** Use transpiler (Babel) for older browser support
- **CSS3 Features:** Graceful degradation for older browsers

9. Troubleshooting

Issue	Cause	Solution
Cannot log in after signup	LocalStorage not enabled	Check browser privacy settings, enable LocalStorage
Workout programs not displaying	JavaScript disabled	Enable JavaScript in browser settings
Calorie calculator shows incorrect result	Invalid input format	Ensure weight, height, age are positive numbers
Page redirects to login unexpectedly	Session expired or browser cache cleared	Re-login with credentials
Test.js fails to run	Node.js not installed	Install Node.js from nodejs.org

10. Future Enhancements

Potential features for project expansion:

1. Backend integration for secure data storage
2. User dashboard with workout progress tracking
3. Mobile app version using React Native or Flutter
4. Social features: friend connections, shared workouts
5. Advanced analytics: calorie history, weight tracking
6. AI-powered personalized program recommendations
7. Integration with wearable devices for activity tracking
8. Community workout sharing and ratings
9. Nutrition database and meal planning
10. Progressive overload tracking and auto-adjustment

