Shift Management System - Complete Documentation

Table of Contents

JavaScript Files Documentation

menu.js

Purpose: Manages the navigation menu, user authentication display, and signout functionality across all pages.

Global Variables

- `currentUser` - Retrieved from localStorage, stores the username of the logged-in user

Functions

Main Execution Block (Not in a function)

- Retrieves current user from localStorage
- Updates the user greeting element to show "Hello [username]!" or "Hello Guest!"
- Handles signout link click event
- Fixes the Shifts link to redirect to homepage.html

Signout Handler (Anonymous function)

- Event listener for signout link click
- Prevents default link behavior with `e.preventDefault()`
- If no user logged in: redirects to login.html

- If user logged in: removes currentUser from localStorage, shows alert, redirects to login

---

login.js

Purpose: Handles user login functionality with validation and error display.

Functions

Main DOMContentLoaded Event Handler

- Initializes all login functionality when page loads
- Creates error message element dynamically
- Sets up form submission handler
- Sets up input event listeners to hide errors

showError(message)

- Displays error message in red text
- Clears input fields
- Focuses back on username field

hideError()

- Hides error message element
- Clears error text content

Form Submit Handler (Anonymous function)

- Prevents default form submission
- Validates username and password are not empty
- Checks if user exists in localStorage
- Verifies password matches stored password
- On success: stores currentUser in localStorage and redirects to homepage
- On failure: shows appropriate error message

---

signup.js

Purpose: Handles new user registration with comprehensive real-time validation.

Functions

Main Initialization Block

- Creates error message elements for each input field
- Sets up all validation event listeners
- Handles form submission

validateEmail(email)

- Checks email format requirements:
    - Cannot start or end with @
    - Must contain exactly one @
    - Domain must contain a period
    - Period cannot be at start or end of domain
- Returns error message or empty string

validateUsername(username)

- Checks minimum 6 characters
- Verifies username uniqueness in localStorage
- Returns error message or empty string

validatePassword(password)

- Checks minimum 8 characters
- Requires at least one letter
- Requires at least one number
- Returns error message or empty string

validateConfirmPassword(confirmPassword, password)

- Ensures passwords match
- Returns error message or empty string

validateFirstName(firstName)

- Checks minimum 2 characters
- Only allows alphabetic characters
- Returns error message or empty string

validateLastName(lastName)

- Same validation as firstName
- Returns error message or empty string

validateAge(age)

- Must be between 18 and 65
- Must be a valid number
- Returns error message or empty string

showError(inputId, message)

- Shows/hides error message for specific input
- Updates error div display and text

hideError()

- Hides all error messages

Real-time Validation Event Listeners

- Each input has an 'input' event listener
- Validates on every keystroke
- Updates error display immediately

Form Submit Handler

- Validates all fields
- Creates user object with all data
- Stores in localStorage under users object
- Clears form on success
- Shows success alert and redirects to login

---

homepage.js

Purpose: Displays user shifts in a table with filtering, editing, and deletion capabilities.

Global Variables

- `shiftsTable` - Reference to the shifts table element
- `filterForm` - Reference to the filter form
- `branchSelect` - Branch filter dropdown
- `positionInput` - Position filter text input
- `totalWageDisplay` - Element showing total wages
- `userShifts` - Array storing current user's shifts

Functions

initializePage()

- Main initialization function
- Logs debug information
- Loads shifts from localStorage
- Sets up filter functionality

loadShifts()

- Clears existing table rows
- Checks user authentication
- Retrieves shifts from localStorage
- Sorts shifts by date (newest first)
- Calls displayShifts() to render

displayShifts(shiftsToDisplay)

- Clears table except header
- Creates table rows for each shift
- Formats date to DD/MM/YYYY
- Adds Delete and Update buttons
- Calculates and displays total wage
- Shows "no shifts" message if empty

displayNoShiftsMessage(message)

- Shows centered italic message in table
- Used for empty states

updateTotalWage(total)

- Updates the total wage display
- Formats to 2 decimal places

deleteShift(shiftId)

- Shows detailed confirmation dialog
- Removes shift from localStorage
- Reloads shifts table
- Shows success alert

updateShift(shiftId)

- Stores shift ID in localStorage
- Redirects to update-shift.html

setupFilters()

- Attaches click event to filter button
- Prevents form submission default

applyFilters()

- Gets selected branch and position text
- Filters userShifts array
- Applies both filters if present
- Redisplays filtered results
- Logs filter operations for debugging

---

update-shift.js

Purpose: Handles adding new shifts and updating existing shifts with validation and overlap detection.

Global Variables

- Form element references (date, times, wage, position, branch)
- `isUpdating` - Boolean flag for edit mode
- `currentShiftData` - Stores shift being edited
- `shiftToUpdateId` - ID of shift being updated

Functions

initializePage()

- Main initialization function
- Gets all form elements with fallback selectors
- Checks user authentication
- Creates message elements
- Loads shift data if updating
- Sets up form submission handler
- Adds cancel button in update mode

calculateShiftWage(startTime, finishTime, hourlyWage)

- Calculates total wage for shift
- Handles overnight shifts (end time < start time)
- Returns wage as string with 2 decimal places

checkShiftOverlap(date, startTime, finishTime, excludeShiftId)

- Prevents double-booking of shifts
- Converts times to minutes for comparison
- Handles overnight shifts
- Returns overlap status and conflicting shift
- Excludes current shift when updating

showError(message) / showSuccess(message)

- Display error or success messages
- Uses paragraph elements below form
- Hides opposite message type

hideMessages()

- Clears both error and success messages

Form Submit Handler

- Validates all required fields
- Checks hourly wage is positive
- Validates position has at least 1 character
- Checks for shift overlaps
- Calculates total wage

- Creates/updates shift object
- Saves to localStorage
- Shows appropriate success message
- Handles both add and update modes

Cancel Button Handler (Update mode only)

- Clears update flag from localStorage
- Redirects to homepage

---

update-profile.js

Purpose: Allows users to update their profile information with validation matching signup requirements.

Functions

initializePage()

- Creates error message elements
- Loads current user profile data
- Sets up validation listeners
- Configures form submission

createErrorMessages()

- Creates error divs for each input field
- Positions below inputs
- Styles with red text

loadUserProfile()

- Retrieves user data from localStorage
- Pre-fills form fields (except passwords)
- Adds username display (read-only)
- Shows username with "(cannot be changed)" note

Validation Functions Same as signup.js but with modifications:

- **validatePassword()** - Optional, empty allowed (keeps existing)
- **validateConfirmPassword()** - Only required if password is being changed
- All other validators identical to signup.js

showError(inputId, message)

- Shows/hides specific error message
- Updates error div for given input

setupValidation()

- Attaches input event listeners
- Provides real-time validation feedback
- Revalidates dependent fields (passwords)

setupFormSubmission()

- Validates all fields on submit
- Preserves existing password if not changed
- Updates user object in localStorage
- Shows success message for 3 seconds
- Clears password fields after update

---

resume-update.js

Purpose: Manages education and occupation records with full CRUD operations.

Global Variables

- Form element references for education and occupation
- `editingEducationId` - ID of education record being edited
- `editingOccupationId` - ID of occupation record being edited
- Message element references

Functions

initializePage()

- Checks user authentication
- Loads existing records
- Sets up form handlers
- Sets maximum year validation

Error/Success Message Functions

- `showEduError()`, `showEduSuccess()`, `hideEduMessages()`
- `showWorkError()`, `showWorkSuccess()`, `hideWorkMessages()`
- Handle message display for each section independently

loadEducationRecords()

- Retrieves education data from localStorage
- Creates HTML table with headers
- Displays each record with Edit/Delete buttons
- Shows truncated descriptions (100 chars)
- Adds expandable rows for long descriptions
- Shows empty state message if no records

loadOccupationRecords()

- Similar to loadEducationRecords but for work experience
- Creates table with workplace, role, years
- Handles expandable descriptions
- Shows empty state message

setupEducationForm()

- Form submission handler for education
- Validates required fields (start year, institution, degree)
- Validates end year > start year
- Creates/updates education record
- Saves to localStorage
- Refreshes display

setupOccupationForm()

- Form submission handler for occupation
- Validates required fields (start year, workplace, role)
- Validates year logic
- Creates/updates occupation record
- Saves to localStorage
- Refreshes display

editEducation(id) / editOccupation(id)

- Populates form with existing record data
- Sets editing mode flag
- Changes button text to "Update"
- Scrolls form into view

deleteEducation(id) / deleteOccupation(id)

- Shows detailed confirmation dialog
- Removes record from localStorage
- Refreshes display
- Shows success message

HTML Files Documentation

login.html

Structure:

- Simple centered login form

- Two input fields: username and password
- Two buttons: Login (submit) and Register (link to signup)
- Uses flexbox for layout
- Includes login.js for functionality

Key Elements:

- Form wraps all inputs
- login-form div for styling container
- Flex break div for layout control
- Register button wrapped in anchor tag

---

signup.html

Structure:

- Centered registration form
- Seven input fields for user data
- Single submit button for registration
- Link to login page for existing users

Key Elements:

- All inputs have specific IDs for validation
- Email input with type="email"
- Password fields with type="password"
- Age input with type="number"
- Button container for alignment
- Uses signup-form class for styling

---

homepage.html

Structure:

- Header with logo and welcome message
- Hamburger menu sidebar with navigation
- Filter form with branch dropdown and position search
- Shifts table with 9 columns
- Add shifts button links to update-shift.html

Key Elements:

- Menu toggle checkbox for sidebar
- Filter form with branch select and job position input
- Table uses for headers
- Total wage display paragraph

- Add shifts wrapped in anchor tag

Notable Features:

- Responsive sidebar navigation
- Table-based shift display
- Real-time filtering capability

---

update-shift.html

Structure:

- Header with navigation menu
- Single form for shift details
- Six input fields for shift information
- Branch dropdown with Israeli cities
- Message paragraphs for errors/success

Key Elements:

- Date and time inputs with proper types
- Number input for hourly wage with step="0.01"
- Select dropdown for branches
- Error and success message paragraphs
- signup-form class reused for consistent styling

---

update-profile.html

Structure:

- Navigation header with menu
- Profile update form
- Six input fields (email, passwords, name, age)
- Single update button

Key Elements:

- Password fields optional (placeholder indicates this)
- Age input with min/max attributes
- update-form class for styling
- Reuses navigation structure

---

resume-update.html

Structure:

- Two separate forms: Education and Occupation
- Each form in its own fieldset
- Container divs for displaying records
- Expandable tables for saved records

Key Elements:

- Fieldsets with legends for visual grouping
- Textarea for descriptions
- Number inputs for years with min/max
- Select dropdown for degree types
- Separate containers for record display
- Message paragraphs for each section

Notable Features:

- Independent form submission for each section
- Table-based record display
- Expandable description rows

CSS Files Documentation

general.css

Purpose: Base styles used across all pages

Key Styles:

- **Body:** Flexbox centered layout, 100vh height, Arial font
- **Form:** Flex layout with center alignment
- **Input:** 10px padding, gray border, 4px radius
- **Button:** Blue (#007bff) background, white text, hover darkens
- **Menu System:**
    - Hidden checkbox for toggle
    - Fixed position sidebar (-250px hidden, 0 visible)
    - Blue theme for navigation
    - Smooth transitions (0.3s)
- **User Greeting:** Bold white text on darker blue background

---

login.css

Purpose: Specific styles for login page

Key Styles:

- **login-form:** Flexbox row layout, max-width 400px
- **Inputs:** Min-width 150px, max-width 200px
- Centers form elements with gaps
- Maintains compact layout

---

signup.css

Purpose: Styles for registration form

Key Styles:

- **signup-form:** Column flex direction, stretch alignment
- **Inputs:** 100% width, 300px minimum
- **Button Container:** Flex layout for button alignment
- **Login Link:** Blue color, underline on hover
- Vertical form layout

---

homepage.css

Purpose: Styles for main dashboard

Key Styles:

- **Header:** Centered content with logo
- **Content Area:** Column flex, centered items
- **Filter Form:** Flex row with gaps, wraps on small screens
- **Shifts Table:**
    - Full width, collapsed borders
    - Blue headers (#007bff)
    - White text on headers
    - Padding and borders for cells
- **Responsive Design:** Adapts to screen size

---

update-shift.css

Purpose: Styles for shift management form

Key Styles:

- **signup-form class:** Reused for consistency
- **Inputs/Select:** 100% width, proper padding
- **Message Container:** Centered text, min-height 30px
- Same button styling as signup

---

update-profile.css

Purpose: Profile update form styles

Key Styles:

- **update-form:** Similar to signup-form
- **Inputs:** Full width with box-sizing
- Consistent with signup page styling
- Maintains form field alignment

---

resume-update.css

Purpose: Styles for resume management

Key Styles:

- **Body:** Auto height, min-height 100vh
- **Fieldsets:** Blue border, gray background, rounded corners
- **Legend:** Bold blue text
- **Inputs/Textareas:** Full width, consistent padding
- **Records Container:**
    - White background with shadow
    - Rounded corners
    - Table styling with blue headers
- **Message Styles:** Red/green backgrounds for errors/success
- **Responsive:** Overflow handling for mobile

Key Design Patterns

Authentication Flow

1. User registers via signup.html
2. Data stored in localStorage under 'users' object
3. Login validates against stored data
4. CurrentUser stored in localStorage
5. Menu.js manages session across pages

Data Storage Structure

```
// Users
users = {
  "username": {
    email: "user@email.com",
    password: "password123",
    firstName: "John",
    lastName: "Doe",
    age: 25
  }
}

// Shifts
shifts = {
  "username": [
    {
      id: "timestamp",
      date: "2024-01-15",
      startTime: "09:00",
      finishTime: "17:00",
      hourlyWage: 25,
      position: "Manager",
      branch: "Tel Aviv",
      totalWage: 200.00
    }
  ]
}

// Resumes
resumes = {
  "username": {
    education: [...],
    occupation: [...]
  }
}
```

Validation Strategy

- Real-time validation on input
- Error messages below fields
- Form submission prevention on errors
- Visual feedback (red errors, green success)
- Consistent validation rules across signup and profile update

UI/UX Patterns

- Blue (#007bff) as primary color
- Consistent button styling
- Responsive sidebar navigation
- Table-based data display
- Confirmation dialogs for destructive actions
- Success messages with auto-dismiss
- Smooth scrolling to edited forms

Security Considerations

1. Passwords stored in plain text (development only)
2. No server-side validation
3. Data persists in localStorage (clearable by user)
4. No encryption or hashing
5. Client-side only authentication

Browser Compatibility

- Modern browsers supporting ES6
- LocalStorage API required
- Flexbox and CSS3 features used
- No IE support

End of Documentation