Automation Assignment Tracker

A comprehensive UI framework for tracking and managing automation testing assignments with support for UI and API testing workflows.

📋 Overview

This project provides an interactive web-based interface for managing automation testing assignments across three use cases:

- **Use Case 1**: Message Box Task (UI Automation)
- **Use Case 2**: Form with Upload Flow (UI Automation)
- **Use Case 3**: Learning Instance API Flow (API Automation)

🚀 Features

- **Interactive Progress Tracking**: Track completion of steps for each use case
- **Test Data Management**: Input and manage test data for all use cases
- **Form Builder Canvas**: Visual form building with drag-and-drop simulation
- **API Testing Interface**: Login authentication and instance creation workflow
- **Validation Results Display**: Real-time display of API validation results
- **Overall Progress Dashboard**: View progress across all use cases

🛠️ Framework and Tools Used

UI Framework

- **React 18+**: JavaScript library for building user interfaces
- **React Hooks**: useState for state management
- **JSX**: JavaScript XML for component structure

Styling & UI

- **Tailwind CSS**: Utility-first CSS framework for styling
- **Lucide React**: Icon library for UI elements
- **Gradient Backgrounds**: Custom color schemes for visual appeal

Testing Framework Support

- **Playwright**: Recommended for cross-browser automation testing
- **Cypress**: Alternative for end-to-end UI testing
- **Page Object Model (POM)**: Design pattern for organizing test code

Development Tools

- **Node.js**: Runtime environment (v16 or higher)
- **npm/yarn**: Package managers
- **ES6+**: Modern JavaScript features

📦 Installation

Prerequisites

- Node.js (v16 or higher)
- npm or yarn

Setup Steps

1. Clone the repository:

```bash
git clone <repository-url>
cd automation-assignment-tracker
```

2. Install dependencies:

```bash
npm install
# or
yarn install
```

3. Start the development server:

```bash
```

```
npm start
# or
yarn start
```

4. Open your browser and navigate to:

```
http://localhost:3000
```

📁 Project Structure

```
automation-assignment-tracker/
├── src/
│   ├── components/
│   │   └── AutomationAssignmentUI.jsx    # Main component
│   ├── App.js                            # Root application
│   └── index.js                          # Entry point
├── public/
├── package.json
└── README.md
```

🎯 Use Cases

Use Case 1: Message Box Task (UI)

Steps:

1. Log in to the application
2. Navigate to Automation from the left-hand menu
3. Click on the Create dropdown and select Task Bot
4. Fill in all mandatory details and click Create
5. Search for Message Box in Actions panel and add it
6. Verify UI element interactions
7. Save the configuration

Test Data Required:

- Bot Name
- Description
- Message Box Title

- Message Box Content

Use Case 2: Form with Upload Flow (UI)

Steps:

1. Log in to the application
2. Navigate to Automation menu
3. Create a new Form
4. Drag and drop Textbox and File Upload elements
5. Verify UI interactions
6. Enter text and upload document
7. Save and verify upload status

Test Data Required:

- Form Name
- Form Description
- Textbox Label
- Textbox Value
- File to Upload

Interactive Features:

- Add/Remove form elements
- Real-time input simulation
- Form save with success confirmation

Use Case 3: Learning Instance API Flow (API)

Steps:

1. Perform API login with credentials
2. Navigate to learning instance under AI tab
3. Create a Learning Instance
4. Validate the created instance

Test Data Required:

- Username

- Password
- Learning Instance Name
- Instance Description

Validation Results Displayed:

- HTTP Status Code (201 Created)
- Response Time
- Response Body Schema (ID, Name, Status)
- Functional Accuracy Checks

💻 Usage

Tracking Progress

1. Click on any use case section to expand it
2. Click on individual steps to mark them as complete
3. View progress bars for each use case
4. Monitor overall progress in the dashboard

Entering Test Data

1. Expand the relevant use case section
2. Fill in the required test data fields
3. Use the interactive features (Form Builder, API Login)
4. View validation results in real-time

Form Builder (Use Case 2)

1. Click "Add Textbox" to add text input fields
2. Click "Add File Upload" to add file upload controls
3. Enter values in the added elements
4. Click "Save Form" to simulate form submission
5. View success confirmation message

API Testing (Use Case 3)

1. Enter username and password

2. Click "Perform API Login"
3. Enter Learning Instance details
4. Click "Create Learning Instance"
5. View comprehensive validation results

🔧 Configuration

Environment Variables

No environment variables required for the UI framework.

Environment/Configuration Notes

System Requirements

- **Operating System**: Windows 10+, macOS 10.15+, or Linux
- **Node.js**: Version 16.x or higher
- **npm**: Version 8.x or higher (or yarn 1.22+)
- **Browser**: Chrome, Firefox, Safari, or Edge (latest versions)
- **Memory**: Minimum 4GB RAM recommended
- **Disk Space**: At least 500MB free space

Dependencies Installation

```json
{
  "dependencies": {
    "react": "^18.2.0",
    "react-dom": "^18.2.0",
    "lucide-react": "^0.263.1"
  },
  "devDependencies": {
    "tailwindcss": "^3.3.0",
    "autoprefixer": "^10.4.14",
    "postcss": "^8.4.24"
  }
}
```

Port Configuration

- Default development port: `3000`

- To change port: Set environment variable `PORT=<port_number>`

Browser Compatibility

- Chrome/Edge: Version 90+
- Firefox: Version 88+
- Safari: Version 14+

Application Configuration

- **State Management**: In-memory state (no backend required)
- **Data Persistence**: Session-based (data clears on page refresh)
- **API Simulation**: Mock API responses for demonstration

Customization Options

- **Color Themes**: Modify Tailwind classes in component
- **Progress Tracking**: Stored in React state (localStorage can be added)
- **Form Elements**: Easily extensible for additional input types

Performance Notes

- Lightweight application (~2MB bundle size)
- Fast rendering with React virtual DOM
- No external API calls required
- Optimized for modern browsers

Customization

To customize the UI:

- Modify color schemes in `className` props
- Update step descriptions in the component arrays
- Adjust validation logic in handler functions
- Add new form elements to the canvas builder
- Extend API validation checks

Integration with Test Automation

This UI framework can be used alongside:

- **Playwright Test Scripts**: Export test data to Playwright fixtures
- **Cypress Test Scripts**: Use test data in Cypress commands
- **Page Object Models**: Map UI elements to POM classes

📊 Features in Detail

Progress Tracking

- Each use case has its own progress bar
- Individual step completion tracking
- Overall progress dashboard showing all three use cases

Data Entry

- Form validation for required fields
- Placeholder text for guidance
- Real-time state management

Interactive Canvas (Use Case 2)

- Dynamic element addition/removal
- Inline value editing
- Save functionality with confirmation

API Validation Display (Use Case 3)

- Status code verification
- Response time measurement
- Schema validation display
- Functional accuracy checks

🧪 Testing Framework Integration

This UI can be integrated with:

- **Playwright**: For end-to-end testing
- **Cypress**: For UI automation

The interface provides test data management that can be exported for use in automated test scripts.

📤 Submission Guidelines

Submit your automation code via:

- GitHub repository, or
- Zip file

Ensure your submission includes:

- Complete test automation code
- This README.md file
- Setup and execution instructions
- Framework and tools documentation

🤝 Contributing

1. Fork the repository
2. Create a feature branch (`git checkout -b feature/improvement`)
3. Commit your changes (`git commit -am 'Add new feature'`)
4. Push to the branch (`git push origin feature/improvement`)
5. Create a Pull Request

📝 License

This project is licensed under the MIT License.

👥 Authors

- Development Team

📞 Support

For issues or questions:

- Create an issue in the repository
- Contact the development team

🔄 Version History

- **v1.0.0** (Current)
    - Initial release

- Three use case implementations
- Interactive form builder
- API testing interface
- Progress tracking dashboard

🎓 Learning Resources

- Playwright Documentation
- Cypress Documentation
- React Documentation
- Tailwind CSS Documentation

Note: This is a UI framework for tracking automation assignments. Actual test automation implementation should be done separately using Playwright or Cypress based on the use cases outlined above.