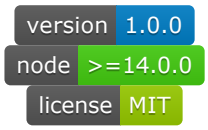
🚀 Government Services Automation Portal



An intelligent automation platform for government services including Vehicle Registration (VAHAN), Passport Services, and E-ID Management with AI-powered natural language processing.

Features • Installation • Usage • API Documentation • Project Structure

---

📋 Table of Contents

🎯 Overview

The **Government Services Automation Portal** is a comprehensive web application that automates various government service workflows using browser automation (Playwright) and AI-powered natural language processing. The platform enables users to interact with government portals through a

conversational interface, making complex form submissions and data retrieval tasks simple and intuitive.

The Why

Problem Statement Indian citizens face a fragmented, complex digital government landscape. Critical services like income tax filing, vehicle registration, document retrieval, and passport services are scattered across multiple unrelated portals, each with its own:

- Navigation patterns
- Authentication mechanisms
- Technical jargon
- Form layouts and validation rules

This fragmentation creates barriers to access, especially for non-technical users, leading to:

- **Wasted Time:** Citizens spend hours navigating multiple websites
- **High Error Rates:** Complex forms lead to mistakes and rejections
- **Accessibility Issues:** Not user-friendly for elderly or less tech-savvy citizens
- **Information Silos:** No unified dashboard to track all government interactions

The Vision

Government-Automate democratizes access to government services by creating a **single, intelligent, conversational interface** that:

1. **Understands Intent:** Natural language processing to recognize what citizens need
2. **Automates Complexity:** Browser automation handles portal navigation
3. **Provides Real-Time Feedback:** Live progress tracking on all tasks
4. **Ensures Security:** Multi-factor authentication and encrypted communications
5. **Remains Transparent:** Citizens see exactly what automation is doing

The Impact

In a few years, we envision Government-Automate like architectures becoming the **de facto standard** for citizen-government interactions in India, serving millions of citizens across all states and enabling:

- **50% reduction** in average time to complete government tasks
- **80% reduction** in form submission errors
- **Universal access** regardless of technical literacy
- **Completely transparent** government service automation

Key Capabilities

- 🤖 **AI-Powered Intent Recognition**: Natural language understanding for task automation
- 🚗 **Vehicle Services**: Search, register, transfer, and update vehicle information
- 🛂 **Passport Services**: Apply for fresh passport applications
- 🆔 **E-ID Management**: Register, search, and update electronic identity records
- 🔐 **Secure Authentication**: JWT-based authentication system
- 📸 **Screenshot Capture**: Visual feedback during automation processes
- 🎨 **Modern UI/UX**: Responsive, gradient-based design with smooth animations

---

✨ Features

Core Features

- **Natural Language Processing**: Users can interact with the system using plain English
- **Multi-Service Support**: Handles VAHAN, Passport, and E-ID services seamlessly
- **Browser Automation**: Headless browser automation using Playwright
- **Real-time Feedback**: Live status updates and screenshot capture during automation
- **Session Management**: Secure session handling for multi-step processes
- **CAPTCHA Handling**: Interactive CAPTCHA resolution workflow
- **Responsive Design**: Mobile-friendly interface with modern UI components

Automation Capabilities

Vehicle Services (VAHAN)

- ✅ Search vehicle details by registration number
- ✅ Register new vehicles
- ✅ Transfer vehicle ownership
- ✅ Update contact information

Passport Services

- ✅ Apply for fresh passport

E-ID Services

- ✅ Register for Electronic ID
- ✅ Search E-ID records
- ✅ Update E-ID information

🔍 Problem Statement

The Current State

Indian government digital services exist in a **fragmented ecosystem**:

```
Citizen ──┬──> Income Tax Portal (Complex UI)
          ├──> VAHAN Portal (Different flow)
          ├──> DigiLocker (Separate auth)
          ├──> Passport Seva (Unique interface)
          ├──> EPFO Portal (Another ecosystem)
          └──> Aadhar Portal (Yet another flow)

Result: Confused citizens, wasted time, high error rates
```

Key Issues Identified

Issue	Impact	Severity
Portal Fragmentation	Citizens must navigate multiple websites	CRITICAL
Complex Authentication	Different login methods across portals	HIGH
Form Complexity	Multiple validation rules, unclear instructions	HIGH
No Progress Tracking	Users don't know status of submitted forms	MEDIUM
Poor Mobile Experience	Many portals not optimized for mobile	MEDIUM
Technical Barriers	Elderly/non-technical users struggle	CRITICAL

The Government-Automate Solution

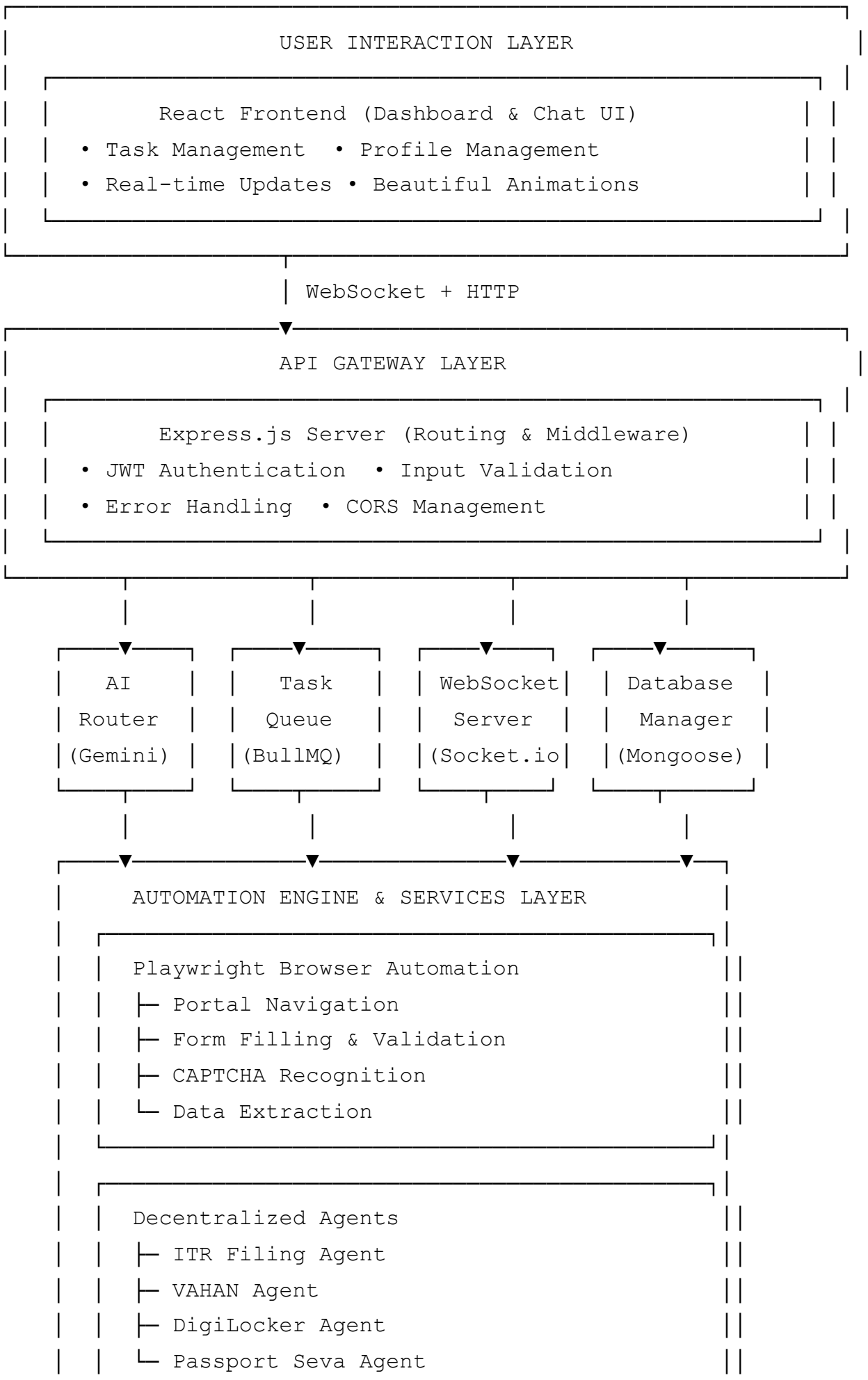
Consolidate all services into:

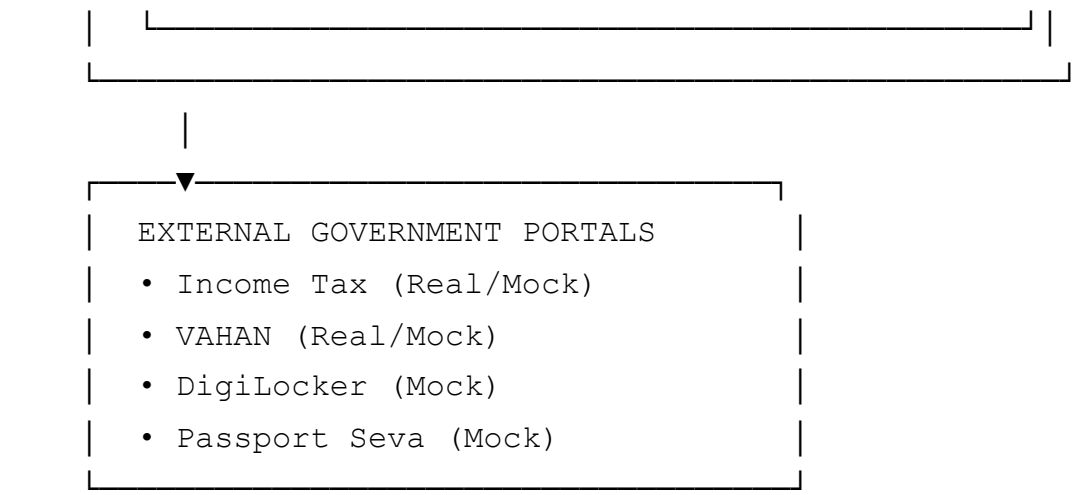
- **One Interface:** Chat-based UI for all interactions
- **One Authentication:** Secure, unified login
- **One Dashboard:** All government tasks in one place
- **One Standard:** Consistent user experience across all services

---

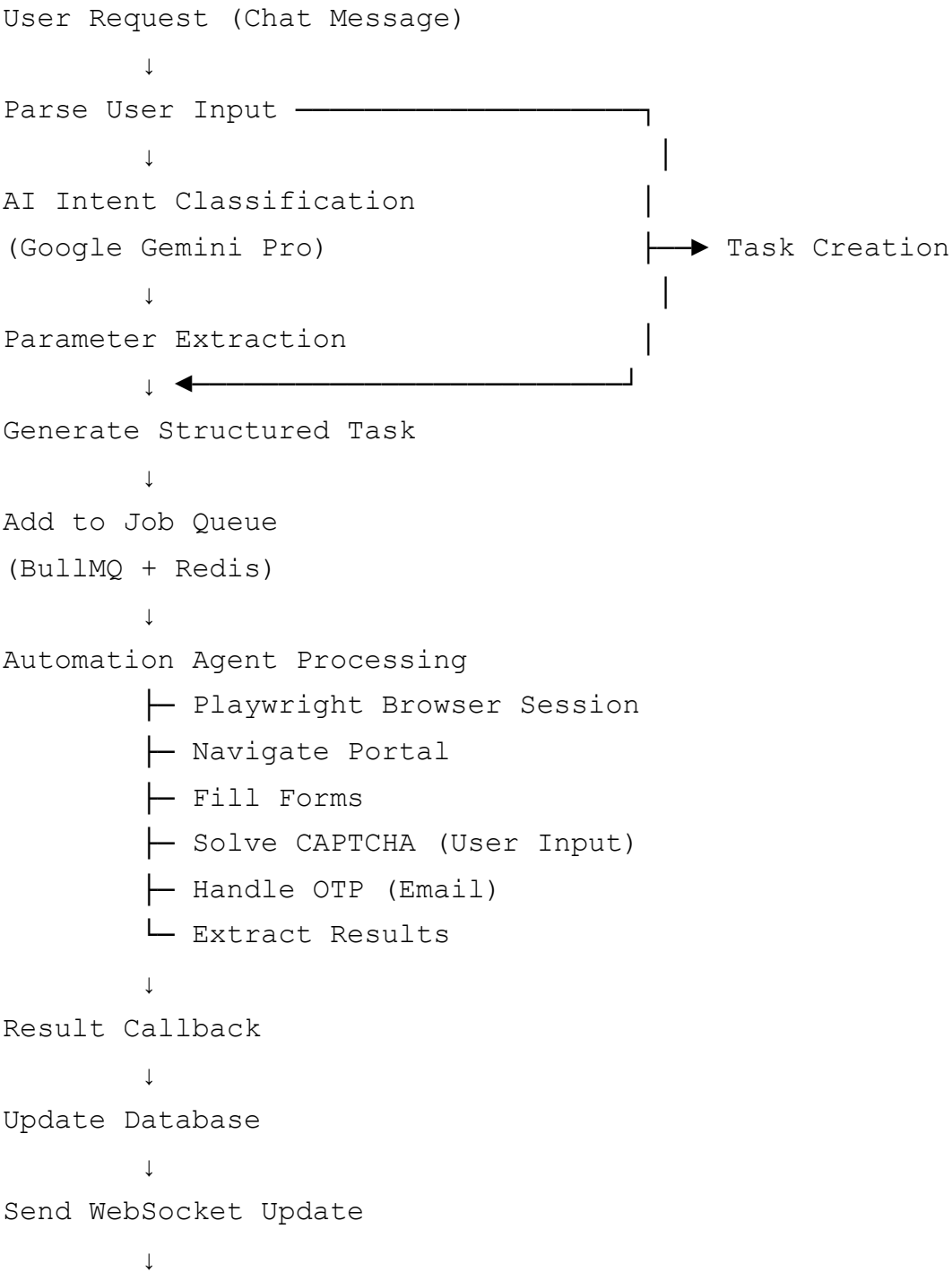
💡 Solution Architecture

High-Level Overview





Data Flow Architecture



```
User Receives Result
(Chat UI)
```

Tech Stack

Backend

Technology	Version	Purpose
Node.js	≥14.0.0	Runtime environment
Express.js	^4.21.2	Web framework
MongoDB	Latest	Primary database
Mongoose	^8.19.3	ODM for MongoDB
Playwright	^1.56.1	Browser automation
Google Generative AI	^0.24.1	Natural language processing
JWT	^9.0.2	Authentication tokens
bcryptjs	^3.0.3	Password hashing
CORS	^2.8.5	Cross-origin resource sharing

Frontend

- **HTML5** - Semantic markup
- **CSS3** - Modern styling with CSS variables and gradients
- **Vanilla JavaScript** - No framework dependencies
- **Inter Font** - Google Fonts typography

Infrastructure

- **MongoDB Atlas** - Cloud database hosting
- **Environment Variables** - Secure configuration management

---

📁 Project Structure

```
my-project/
|
```

```
├── backend/                          # Backend application
│   ├── automation/                   # Playwright automation scripts
│   │   ├── searchVehicle.js          # Vehicle search automation
│   │   ├── registerVehicle.js        # Vehicle registration automation
│   │   ├── transferOwnership.js      # Ownership transfer automation
│   │   ├── updateContacts.js         # Contact update automation
│   │   ├── freshPassport.js          # Passport application automation
│   │   ├── registerEid.js            # E-ID registration automation
│   │   ├── searchEid.js              # E-ID search automation
│   │   ├── updateEid.js              # E-ID update automation
│   │   └── screenshots/              # Captured screenshots directory
│   │
│   ├── config/                       # Configuration files
│   │   └── db.js                     # Database connection configuration
│   │
│   ├── login-setup/                  # Authentication module
│   │   ├── controllers/
│   │   │   └── auth-controller.js    # Authentication controllers
│   │   ├── middleware/
│   │   │   └── auth-middleware.js    # JWT authentication middleware
│   │   ├── models/
│   │   │   └── user.js               # User model
│   │   └── routes/
│   │       └── auth-routes.js        # Authentication routes
│   │
│   ├── middleware/                   # Application middleware
│   │   └── authMiddleware.js         # General auth middleware
│   │
│   ├── models/                       # Database models
│   │   ├── userModel.js              # User data model
│   │   └── vehicleModel.js           # Vehicle data model
│   │
│   ├── routes/                       # API routes
│   │   ├── api.js                    # General API routes
│   │   ├── authRoutes.js             # Authentication routes
│   │   ├── automationRoutes.js       # Automation task routes
│   │   └── brainRoutes.js            # AI/NLP processing routes
│   │
│   ├── utils/                        # Utility functions
│   │   └── sendEmail.js              # Email sending utility
│   │
```

```
│   ├── server.js                    # Main server file
│   ├── package.json                 # Dependencies and scripts
│   └── .env                         # Environment variables (not in repo)
│
└── frontend/                        # Frontend application
    ├── index.html                   # Landing page
    ├── login.html                   # Login page
    ├── register.html                # Registration page
    ├── task-portal.html             # Main automation portal
    ├── task-portal.js               # Portal JavaScript logic
    ├── task-portal-styles.css       # Portal styling
    ├── task-portal-auth.js          # Portal authentication
    ├── eid-mock-portal.html         # E-ID mock portal
    ├── mock-pastport-website.html   # Passport mock portal
    ├── auth-styles.css              # Authentication page styles
    ├── styles.css                   # General styles
    └── script.js                    # General JavaScript
```

📦 Prerequisites

Before you begin, ensure you have the following installed:

- **Node.js** (v14.0.0 or higher) - Download
- **npm** (comes with Node.js) or **yarn**
- **MongoDB Atlas Account** (or local MongoDB instance)
- **Google Generative AI API Key** - Get API Key
- **Git** (for cloning the repository)

🚀 Installation

1. Clone the Repository

```
git clone <repository-url>
cd my-project
```

2. Install Backend Dependencies

```
cd backend
npm install
```

3. Install Playwright Browsers

```
npm run install-playwright
```

This will download the required browser binaries for Playwright automation.

4. Environment Configuration

Create a `.env` file in the `backend/` directory:

```
cd backend
touch .env
```

Add the following environment variables (see Configuration section for details):

```
# Server Configuration
PORT=5000

# MongoDB Configuration
MONGO_URI=mongodb+srv://username:password@cluster.mongodb.net/database?re

# E-ID Database Configuration
EID_MONGO_URI=mongodb+srv://username:password@cluster.mongodb.net/eidData

# JWT Configuration
JWT_SECRET=your-super-secret-jwt-key-here

# Google Generative AI
GOOGLE_AI_API_KEY=your-google-ai-api-key-here

# Email Configuration (Optional)
EMAIL_HOST=smtp.gmail.com
EMAIL_PORT=587
EMAIL_USER=your-email@gmail.com
EMAIL_PASS=your-app-password
```

⚙️ Configuration

Environment Variables

Variable	Description	Required	Default
`PORT`	Server port number	No	`5000`
`MONGO_URI`	MongoDB connection string for main database	Yes	-
`EID_MONGO_URI`	MongoDB connection string for E-ID database	Yes	-
`JWT_SECRET`	Secret key for JWT token signing	Yes	-
`GOOGLE_AI_API_KEY`	Google Generative AI API key	Yes	-
`EMAIL_HOST`	SMTP server host	No	-
`EMAIL_PORT`	SMTP server port	No	`587`
`EMAIL_USER`	SMTP username	No	-
`EMAIL_PASS`	SMTP password	No	-

Database Setup

1. **Main Database**: Create a MongoDB database for user authentication and vehicle data
2. **E-ID Database**: Create a separate MongoDB database for E-ID records

Both databases can be hosted on MongoDB Atlas or locally.

💻 Usage

Starting the Server

Development Mode (with auto-reload)

```
cd backend
npm run dev
```

Production Mode

```
cd backend
npm start
```

The server will start on `http://localhost:5000` (or the port specified in `.env`).

Accessing the Application

1. **Landing Page**: `http://localhost:5000/`
2. **Login Page**: `http://localhost:5000/login.html`
3. **Registration**: `http://localhost:5000/register.html`
4. **Task Portal**: `http://localhost:5000/task-portal.html`
5. **E-ID Mock Portal**: `http://localhost:5000/eid-mock-portal.html`
6. **Passport Mock Portal**: `http://localhost:5000/mock-pastport-website.html`

Using the Task Portal

1. **Login/Register**: Create an account or login with existing credentials
2. **Natural Language Input**: Type your request in plain English
    - Example: "Search vehicle DL01AB1234 from Delhi"
    - Example: "Register for E-ID"
    - Example: "I want to apply for a passport"
3. **Follow Prompts**: The AI will guide you through the process
4. **Complete Forms**: Fill in the required information as prompted
5. **View Results**: See automation results with screenshots

📡 API Documentation

Authentication Endpoints

Register User

```
POST /api/portal-auth/register
Content-Type: application/json

{
  "name": "John Doe",
  "email": "john@example.com",
```

```
  "password": "securePassword123"
}
```

Login

```
POST /api/portal-auth/login
Content-Type: application/json

{
  "email": "john@example.com",
  "password": "securePassword123"
}
```

Get Profile

```
GET /api/portal-auth/profile
Authorization: Bearer <JWT_TOKEN>
```

Automation Endpoints

Execute Automation Task

```
POST /api/automation/execute
Content-Type: application/json
Authorization: Bearer <JWT_TOKEN>

{
  "taskType": "search",
  "sessionId": "optional-session-id",
  "step": "start",
  "data": {
    "registrationNumber": "DL01AB1234",
    "state": "Delhi"
  }
}
```

Available Task Types:

- `search` - Search vehicle
- `register` - Register vehicle

- `transfer` - Transfer ownership
- `update` - Update contacts
- `passport_fresh` - Apply for passport
- `eid_register` - Register E-ID
- `eid_search` - Search E-ID
- `eid_update` - Update E-ID

Handle CAPTCHA Response

```
POST /api/automation/captcha
Content-Type: application/json
Authorization: Bearer <JWT_TOKEN>

{
  "sessionId": "session-id",
  "captchaText": "ABC123"
}
```

AI/NLP Endpoints

Process Natural Language Request

```
POST /api/brain/process
Content-Type: application/json
Authorization: Bearer <JWT_TOKEN>

{
  "message": "Search vehicle DL01AB1234 from Delhi"
}
```

Response:

```
{
  "task": "search",
  "entities": {
    "registrationNumber": "DL01AB1234",
    "state": "Delhi"
  },
  "reply": "I'll help you search for vehicle DL01AB1234 from Delhi..."
}
```

E-ID API Endpoints

Register E-ID

```
POST /api/register
Content-Type: application/json

{
  "name": "John Doe",
  "dob": "1990-01-01",
  "gender": "Male",
  "phone": "9876543210",
  "address": "123 Main St, City"
}
```

Search E-ID

```
GET /api/search/:eId
```

Update E-ID

```
PUT /api/update
Content-Type: application/json

{
  "eId": "123456789012",
  "name": "John Updated",
  "phone": "9876543210",
  "address": "456 New St, City"
}
```

🤖 Automation Scripts

Vehicle Automation Scripts

`searchVehicle.js`

Searches for vehicle details by registration number and state.

Parameters:

- `registrationNumber` (string): Vehicle registration number
- `state` (string): State code (e.g., "Delhi", "Maharashtra")

`registerVehicle.js`

Registers a new vehicle with provided details.

Parameters:

- Vehicle registration details (number, owner, etc.)

`transferOwnership.js`

Transfers vehicle ownership from one person to another.

Parameters:

- Current owner details
- New owner details
- Vehicle registration number

`updateContacts.js`

Updates contact information for a registered vehicle.

Parameters:

- Registration number
- New contact details

Passport Automation Scripts

`freshPassport.js`

Automates the fresh passport application process.

Parameters:

- Personal details
- Address information
- Document uploads

E-ID Automation Scripts

`registerEid.js`

Registers a new E-ID with personal information.

Parameters:

- Name, DOB, Gender
- Phone number
- Address

`searchEid.js`

Searches for E-ID records by E-ID number.

Parameters:

- `eId` (string): 12-digit E-ID number

`updateEid.js`

Updates existing E-ID information.

Parameters:

- E-ID number
- Fields to update (name, phone, address)

🎨 Frontend Portals

Task Automation Portal (`task-portal.html`)

The main interface for interacting with the automation system. Features:

- **Chat Interface**: Natural language conversation with AI
- **Form Handling**: Dynamic form generation based on task type
- **Screenshot Display**: Visual feedback during automation
- **Status Updates**: Real-time progress indicators
- **Modern UI**: Gradient design with smooth animations

Mock Portals

E-ID Mock Portal (`eid-mock-portal.html`)

Simulated E-ID management portal for testing automation scripts.

Passport Mock Portal (`mock-pastport-website.html`)

Simulated passport application portal for testing automation.

🔧 Development

Project Scripts

```
# Start development server with auto-reload
npm run dev

# Start production server
npm start

# Install Playwright browsers
npm run install-playwright
```

Code Structure Guidelines

1. **Backend Routes**: All API routes are organized in `backend/routes/`
2. **Automation Scripts**: Each automation task has its own script in `backend/automation/`
3. **Frontend**: Static HTML/CSS/JS files in `frontend/`
4. **Models**: Database schemas in `backend/models/`
5. **Middleware**: Authentication and other middleware in `backend/middleware/`

Adding New Automation Scripts

1. Create a new script in `backend/automation/`
2. Export the main function
3. Add route handler in `backend/routes/automationRoutes.js`
4. Update `backend/routes/brainRoutes.js` to recognize the new task
5. Add frontend handling in `frontend/task-portal.js`

🐛 Troubleshooting

Common Issues

Port Already in Use

```
# Windows
netstat -ano | findstr :5000
taskkill /PID <PID> /F

# Linux/Mac
lsof -ti:5000 | xargs kill -9
```

MongoDB Connection Error

- Verify `MONGO_URI` in `.env` file
- Check MongoDB Atlas IP whitelist
- Ensure database credentials are correct

Playwright Browser Not Found

```
npm run install-playwright
```

CAPTCHA Timeout

- Increase timeout in automation scripts
- Check network connectivity
- Verify mock portal is accessible

JWT Token Expired

- Re-login to get a new token
- Check `JWT_SECRET` in `.env`

🤝 Contributing

Contributions are welcome! Please follow these guidelines:

1. **Fork the repository**

2. **Create a feature branch** (`git checkout -b feature/amazing-feature`)
3. **Commit your changes** (`git commit -m 'Add some amazing feature'`)
4. **Push to the branch** (`git push origin feature/amazing-feature`)
5. **Open a Pull Request**

Code Style

- Use consistent indentation (2 spaces)
- Follow JavaScript ES6+ conventions
- Add comments for complex logic
- Keep functions focused and modular

📄 License

👥 Authors

- **Ch Pranav Tej (CS24B057)** - *Scrum Master, AIML Dev, Full Stack Dev*- Pranav Tej - Contributions include Management of All Tasks, Organizing meetings, Planning pipeline, Building of ITRFiling script, Creating Income Tax mock portal and the Decentralized Agentic Platform for Income Tax Website Link, and final integration of all scripts.
- **M Nikhil (CS24B026)** - *Full Stack Dev, AIML Dev* - Nikhil - Contributions include Planning Pipeline, Building of Vahan portal, Vahan portal automation scripts, and Centralized Agentic Platform for the 3 websites with full on integration. Also wrote the documentation
- **G Siddhardha (CS24B012)** - *Full Stack Dev, AIML Dev* - Siddhardha - Contributions include Front end Design, Building of E-ID Portal, E-ID portal automation scripts, and Centralized Agentic Platform for the 3 websites with help in integration.
- **M Vinay Sai (CS24B027)** - *Full Stack Dev, AIML Dev* - Vinay - Contributions include building of the Passport Seva portal, Passport Seva portal automation scripts, and help in Centralized Agentic Platform for the 3 websites.
- **S Hemanth (CS24B044)** - *Full Stack Dev, AIML Dev* - Hemanth - Contributions include building of the Passport Seva portal, Backend Design, and help in integration of the final Centralized Agentic Platform for the 3 websites with help in integration.

We have done all commits for the project using Nikhil's github account, i.e., after integrating our files into the framework, we have sent the files to Nikhil in order to push them into this repo.

🙏 Acknowledgments

- **Playwright** - For excellent browser automation capabilities
- **Google Generative AI** - For natural language processing
- **MongoDB** - For robust database solutions
- **Express.js** - For the powerful web framework
- **Redis** - For the amazing queue processing

📞 Support

For support, mail to pranavtej.9.1a@gmail.com, or open an issue in the repository.

Made with ❤️ for Government Services Automation

⭐ Star this repo if you find it helpful!