## 1. Introduction

In the modern digital era, information about events is scattered across multiple platforms, making discovery, tracking, and management inefficient. During my internship, I worked on, a full-stack web application aimed at solving this problem by providing a **centralized event discovery and management platform**.

The project integrates **frontend development, backend APIs, authentication mechanisms, database management, and automation using web scraping**, making it a comprehensive real-world application aligned with industry practices.

---

## 2. Objective of the Project

The primary objectives of the project are:

- To design a **centralized platform** for discovering and managing events
- To automate event data collection using **Python-based web scraping**
- To implement **secure authentication** using Google OAuth 2.0
- To develop a scalable **MERN-based full-stack architecture**
- To follow **industry-level security and Git version control practices**

---

## 3. Problem Statement

Event-related information is often:

- Distributed across multiple websites
- Manually collected and updated
- Lacking proper dashboards and analytics

This results in inefficiency for users, marketers, and businesses.
LOUDERWORLD addresses this problem by **automating data collection** and **presenting it through a unified web interface**.

---

## 4. Scope of the Project

### Included in Scope

- Web-based event discovery system
- Google OAuth authentication
- Backend APIs for event and lead management
- Python scraper using Playwright
- Secure environment variable handling

### Excluded from Scope

- Payment gateway integration
- Mobile application
- AI-based recommendations (future scope)

---

## 5. System Architecture

The system follows a **modular client-server architecture**:

- **Frontend:** React + Vite for fast UI rendering
- **Backend:** Node.js & Express for REST APIs
- **Database:** MongoDB for structured data storage
- **Scraper:** Python + Playwright for automated data collection

**Data Flow:**

```
User → React Frontend → Express Backend → MongoDB
                                ↓
                         Python Scraper
                                ↓
                     External Event Websites
```

---

## 6. Technology Stack Used

### Frontend

- React.js
- Vite
- Tailwind CSS
- Axios

### Backend

- Node.js
- Express.js
- Passport.js (Google OAuth)
- MongoDB with Mongoose

### Automation & Scraping

- Python
- Playwright
- Logging & Error Handling

### Tools & Practices

- Git & GitHub
- dotenv for environment variables
- PM2 (deployment ready)
- ESLint & Prettier

---

## 7. Features Implemented

### 1. Authentication System

- Google OAuth 2.0 integration
- Secure session management
- Protected routes for dashboards

### 2. Event Dashboard

- Centralized event listing
- Lead management system
- Scalable API structure

### 3. Web Scraping Module

- Automated event data extraction
- Playwright-based browser automation
- Structured and modular scraper design

### 4. Security & Version Control

- Sensitive files hidden using `.gitignore`
- Secrets managed via `.env` files
- Clean and professional repository structure

---

## 8. Project Folder Structure

```
LOUDERWORLD/
├── frontend/
│   └── src/
├── backend/
│   ├── routes/
│   ├── controllers/
│   └── config/
├── scraper/
│   └── scripts/
├── .env.example
├── .gitignore
├── README.md
└── requirements.txt
```

---

## 9. Testing & Validation

- API testing using Postman
- Manual UI testing
- Scraper output validation via logs
- OAuth redirect and authentication testing
- Error handling and fallback testing

---

## 10. Limitations

- Scraping depends on third-party website structure
- Anti-bot mechanisms may affect scraper reliability
- No mobile app support currently
- Manual scraper updates required if websites change

---

## 11. Learning Outcomes

Through this internship project, I gained hands-on experience in:

- Full-stack web development (MERN)
- OAuth-based authentication systems
- REST API design
- Web scraping using Playwright
- Secure coding practices
- Git version control & project structuring
- Real-world debugging and deployment readiness

---

## 12. Future Enhancements

- AI-based event recommendation system
- Advanced analytics dashboard
- Role-based access control (RBAC)
- Payment gateway integration
- Mobile application using React Native

---

## 13. Conclusion

The project successfully demonstrates the integration of frontend, backend, database, authentication, and automation technologies into a single scalable system. This internship project helped me understand **real-world software development workflows**, security practices, and production-level project structuring, making it a valuable learning experience.

---

## 14. References

- React Documentation – https://react.dev
- Node.js Documentation – https://nodejs.org
- Express.js – https://expressjs.com
- MongoDB Docs – https://www.mongodb.com/docs
- Playwright – https://playwright.dev
- Google OAuth – https://developers.google.com/identity