🎤 Presentation Speech (Japanese - Romaji Version)

Shain Kanri System no Demonstration

Duration: 12-13 minutes | Audience: Senior Engineers

🎯 Section 1: Opening & Introduction (1 minute)

Kaishi no Aisatsu

Ohayou gozaimasu. Honjitsu wa, watashi no presentation ni ojikan wo itadaki, arigatou gozaimasu.

Watashi no namae wa [Your Name] desu. Genzai, InterSystems IRIS to React.js no gijutsu wo benkyou shite iru intern desu.

Kyou wa, watashi ga kaihatsu shita Kantan na Shain Kanri System wo demonstration itashimasu.

Kono presentation wa mittsu no main area wo cover shimasu:

1. System zentai no workflow no live demonstration
2. React.js to TypeScript de tsukutta frontend architecture
3. InterSystems IRIS database wo tsukatta backend jissou

Zentai de oyoso 12 kara 13 fun no jikan wo itadakimasu. Saigo ni shitsumon wo ukemasu.

Dewa, hajimemashou.

📱 Section 2: Project Overview (1.5 minutes)

Project Shoukai

Kore wa web-based no Shain Kanri System desu. Shain data no kanri lifecycle zentai wo handle shimasu.

[Show browser with application homepage]

System wa modern na full-stack architecture de tsukurarete imasu.

Frontend Technologies:

- React 18 to TypeScript - type safety no tame
- Material-UI - component library toshite

- React Router - single-page application navigation
- Axios - API communication
- Vite - build tool

Backend Technologies:

- InterSystems IRIS database
- ObjectScript - server-side logic
- RESTful API architecture
- SQL - data queries

Shuyou na Kinou: System wa itsutsu no core functionalities wo provide shimasu:

1. User touroku to ninshou
2. Shain sakusei - comprehensive na data fields
3. Shain list - kensaku to filter kinou tsuki
4. Shain henshuu - validation tsuki
5. Soft delete - data hozon

Data Flow: Application wa standard REST architecture ni shitagaimasu. React frontend wa IRIS backend to HTTP API calls de communication shimasu. Subete no data wa JSON format de transmitted saremasu. System wa Nihongo characters no tame ni UTF-8 encoding wo tadashiku handle shimasu.

Dewa, user touroku kara hajimete, kanzen na workflow wo demonstration shimasu.

🔐 Section 3: User Registration Demo (1 minute)

Touroku no Demonstration

Mazu, user touroku process wo omise shimasu.

[Click "アカウント登録" button]

Kore ga touroku form desu. Material-UI TextField components wo tsukatte, React no controlled component pattern de tsukurarete imasu.

Atarashii user wo touroku shimasu:

- Namae: Tanaka Tarou
- Email: tanaka.taro@example.com

- Password: password123

[Fill in the form]

Frontend de, client-side validation wo jissou shite imasu. Tatoeba, email field wa tadashii format wo check shimasu. Soshite password wa sukunakutomo 8 moji hitsuyou desu.

[Click "登録" button]

Touroku button wo click suru to, frontend wa POST request wo `/sem/signup` ni okurimasu.

[Show browser console/network tab]

Network tab de, API call ga miemasu. Request body ni wa, user no jouhou ga JSON format de fukumarete imasu.

Backend de wa, ObjectScript method `AccountRegistration()` ga kono request wo process shimasu. Ikutsuka no operation wo okonaimasu:

1. Input data wo validation shimasu
2. SQL query de, duplicate email address wo check shimasu
3. `tblAccount` class no atarashii instance wo sakusei shimasu
4. `%Save()` method wo tsukatte, IRIS database ni data wo save shimasu

[Point to success message]

Touroku ga seikou shimashita. System wa jidouteki ni sign-in page ni redirect shimasu.

Database de wa, `SEM.tblAccount` table ni atarashii record ga sakusei saremashita.

🔑 Section 4: Login Demo (1 minute)

Ninshou no Demonstration

Dewa, login process wo susumemasu.

[On Sign In page]

Ima touroku shita credentials wo tsukaimasu:

- Email: tanaka.taro@example.com
- Password: password123

[Fill in login form]

[Click "ログイン" button]

Login button wo click suru to, POST request ga `/sem/signin` ni okurareru.

[Show network tab]

Backend no `AccountLogin()` method wa SQL query wo jikkou shimasu:

```sql
SELECT ID, Password FROM SEM.tblAccount WHERE Email = ?
```

Soshite, provided password to stored password wo hikaku shimasu.

Ninshou ga seikou suru to, frontend wa login status wo localStorage ni hozon shimasu. Kore ni wa:

- `isLoggedIn: true`
- `userEmail: tanaka.taro@example.com`

Kore ni yotte, page refresh shitemo session ga persistent shimasu.

[Show successful redirect to Employee List]

Ima, Shain List page ni redirect saremashita. Kono page wa `ProtectedRoute` component ni yotte protected sarete imasu. Kore wa, rendering mae ni authentication status wo check shimasu.

Moshi user ga login sezu ni kono page ni access shiyou to sureba, jidouteki ni sign-in page ni redirect saremasu.

👥 Section 5: Add Employee Demo (1.5 minutes)

Shain Tsuka no Demonstration

Dewa, atarashii shain wo tsuika suru houhou wo demonstration shimasu.

[Click "新規登録" button]

Kono form wa React no `useState` hook wo tsukatte state management wo shite imasu. Kakko no input field wa controlled component desu. Sore wa, value ga React state ni yotte controlled sareru imi desu.

Shain jouhou wo nyuuryoku shimasu:

[Fill in form while speaking]

- Shain ID: 10001 (kore wa choudo 5 keta de nakereba narimasen)
- Namae: Yamada Hanako
- Kana Namae: Yamada Hanako

- Seibetsu: Josei
- Yuubin Bangou: 100-0001
- Juusho: Tokyo-to Chiyoda-ku
- Denwa Bangou: 03-1234-5678
- Busho: Engineering
- Taishoku Flag: Unchecked (genzai koyou chuu)

[Click "登録" button]

Form wa comprehensive na validation wo jissou shite imasu:

- Required field no check
- Shain ID format validation (choudo 5 keta)
- Email format validation, moshi provided sareba

[Show confirmation dialog]

Submit mae ni, Material-UI Dialog component ga kakunin prompt wo display shimasu. Kore wa destructive matawa juuyou na operation no best practice desu.

[Click "はい" to confirm]

[Show network tab]

Frontend wa POST request wo `/sem/employee` ni, shain data wo JSON format de okurimasu.

Backend de, `CreateEmployee()` method wa kono steps wo jikkou shimasu:

1. **JSON request wo parse suru** - `%FromJSON()` wo tsukatte
2. **Required fields wo validation suru** - Shain ID to Namae wa mandatory
3. **Duplicate wo check suru** - kono SQL query de:

```sql
SELECT ID FROM SEM.tblEmployee
WHERE EmployeeId = ? AND deleteFlg = 0
```

4. **Nihongo characters wo handle suru** - UTF-8 conversion:

```objectscript
```

```
$ZCONVERT(requestObject.name, "I", "UTF8")
```

5. **Atarashii shain object wo sakusei suru**:

```objectscript
Set newEmployee = ##Class(SEM.tblEmployee).%New()
```

6. **Properties wo set suru** - fukumu:
   - `deleteFlg = 0` (active shain)
   - `upDateTime = $ZDATETIME($HOROLOG, 3)` (genzai no timestamp)
7. **Database ni save suru** - `%Save()` method de

[Show success message and redirect to list]

Shain ga seikou ni sakusei saremashita. Shain List page wa jidouteki ni refresh shite, atarashii shain wo display shimasu.

🔍 Section 6: Search and View Demo (0.5 minutes)

Shain List no Kinou

Kensaku to filtering kinou wo hayaku demonstration shimasu.

[Show Employee List table]

Table wa subete no active shain wo display shimasu. `deleteFlg = 1` no shain wa default de hidden desu.

[Type "山田" in search box]

Kensaku kinou wa React no `useMemo` hook wo tsukatte, performance optimization no tame ni jissou sarete imasu. Kore wa ID matawa Namae ga search keyword wo fukumu shain wo filter shimasu.

```typescript
const filteredEmployees = useMemo(() => {
  return employees.filter(emp =>
    emp.EmployeeId.includes(searchKeyword) ||
    emp.Name.includes(searchKeyword)
  );
}, [employees, searchKeyword]);
```

Kore wa client-side filter desu. Dakara API call wa hitsuyou arimasen. Filtering wa sokuji browser de okonawaremasu.

[Click on column headers to sort]

Table wa mata column headers wo click suru koto de sorting wo support shimasu. Shain ID matawa Namae de ascending matawa descending order ni sort dekimasu.

[Show pagination controls if applicable]

Ookii dataset no tame ni, pagination wo jissou shite imasu. Default de, 10 rows ga page goto ni displayed saremasu.

✏️ Section 7: Edit Employee Demo (1.5 minutes)

Shain Koushin no Demonstration

Dewa, edit kinou wo demonstration shimasu.

[Click "編集" button on 山田花子's row]

[Show URL change to /employees/42]

URL ga `/employees/:id` ni kawaru no ni kizuite kudasai. React Router no `useParams` hook ga URL kara shain ID wo extract shimasu.

[Show pre-filled form]

Kono page ga load sareru toki, `useEffect` hook ga API call wo trigger shimasu:

```typescript
useEffect(() => {
  if (!isNewEmployee && employeeId) {
    loadEmployeeData(employeeId);
  }
}, [isNewEmployee, employeeId]);
```

Frontend wa GET request wo `/sem/employee/42` ni okurimasu.

Backend no `GetEmployeeById()` method wa jikkou shimasu:

```objectscript
Set sqlQuery = "SELECT * FROM SEM.tblEmployee WHERE ID = ? AND deleteFlg = 0"
```

Response data wa jidouteki ni form fields wo populate shimasu. Kore wa shain wo tsuika suru tame ni tsukatta form component to onaji mono desu ga, "edit mode" desu.

Ima, ikuraka no jouhou wo henkō shimasu:

[Change Department field]

- Busho: Engineering → HR

[Change Phone Number]

- Denwa: 03-1234-5678 → 03-9876-5432

[Click "更新" button]

[Show confirmation dialog]

Mata, kono juuyou na operation no tame ni kakunin dialog wo display shimasu.

[Click "はい" to confirm]

[Show network tab]

Frontend wa PUT request wo `/sem/employee/42` ni, updated data to tomo ni okurimasu.

Backend no `UpdateEmployee()` method:

1. **Sonzai suru shain object wo open suru**:

```objectscript
Set employee = ##class(SEM.tblEmployee).%OpenId(id)
```

2. **ID ga sonzai suru ka validation suru**
3. **Properties wo atarashii values de update suru**
4. **Timestamp wo update suru**: `upDateTime = $ZDATETIME($HOROLOG, 3)`
5. **Henkou wo save suru**: `employee.%Save()`

[Show success message and redirect]

Shain jouhou ga seikou ni updated saremashita. List wa ima modified data wo display shimasu.

Kore wa IRIS no object persistence capabilities wo demonstration shite imasu. `%OpenId()` method wa sonzai suru object wo load shi, watashitachi wa sono properties wo modify shi, `%Save()` ga henkou wo persist shimasu.

🗑️ Section 8: Delete Employee Demo (0.5 minutes)

Soft Delete no Demonstration

Saigo ni, delete kinou wo demonstration shimasu.

[Click "削除" button on 山田花子's row]

[Show delete confirmation dialog]

Kono system no juuyou na aspect wa, watashitachi ga soft delete wo jissou shite iru koto desu. Hard delete de wa arimasen.

[Click "はい" to confirm deletion]

[Show network tab]

Frontend wa DELETE request wo `/sem/employee/42` ni okurimasu.

Backend no `DeleteEmployee()` method:

1. **Shain object wo open suru**: `%OpenId(id)`
2. **Delete flag wo set suru**: `employee.deleteFlg = 1`
3. **Timestamp wo update suru**: `upDateTime = $ZDATETIME($HOROLOG, 3)`
4. **Henkou wo save suru**: `employee.%Save()`

[Show employee disappears from list]

Shain wa ima list kara removed saremashita. Shikashi, record wa mada database ni `deleteFlg = 1` de sonzai shite imasu.

Soft delete no merits:

- Audit trails no tame no data hozon
- Data retention regulations to no compliance
- Deleted records wo restore suru nouryoku
- Referential integrity wo maintain suru

Hontou ni record wo delete suru tame ni wa, chokusetsu database access matawa betsu no purge operation ga hitsuyou desu.

🏗️ Section 9: System Architecture Explanation (2 minutes)

Architecture no Gaiyou

Dewa, zentai no system architecture wo motto kuwashiku setsumei shimasu.

[Can show architecture diagram if prepared]

Frontend Architecture

Frontend wa React 18 to TypeScript de tsukurareta Single Page Application desu.

Project Kouzou:

```
src/
├── pages/          # Page components (SignIn, SignUp, EmployeeList, EmployeeDetail)
├── components/     # Reusable components (Layout, ProtectedRoute)
├── services/       # API communication layer (api.ts)
├── types/          # TypeScript type definitions
└── utils/          # Utility functions (auth.ts)
```

State Management: Watashitachi wa React Hooks wo moppara tsukaimasu:

- `useState` - component state no tame
- `useEffect` - side effects to data fetching no tame
- `useMemo` - computed values to performance no tame
- `useParams` to `useNavigate` - routing no tame

Redux matawa Context API wo tsukaimasen. Naze nara, application scale ga global state management wo hitsuyou to shinai kara desu. Authentication state wa localStorage ni persist sarete imasu.

Shuyou na Design Patterns:

1. **Controlled Components** - Form inputs wa React state ni yotte controlled sareru
2. **Protected Routes** - Authentication guards no tame no HOC pattern
3. **Service Layer** - API calls wa betsu no service ni abstracted sareru
4. **Type Safety** - Subete no data structures no tame no TypeScript interfaces

Backend Architecture

Backend wa ObjectScript to InterSystems IRIS wo tsukaimasu.

Database Schema:

Futatsu no main persistent classes ga arimasu:

1. tblAccount (Ninshou)

```objectscript
Class SEM.tblAccount Extends %Persistent
{
    Property Email As %String(MAXLEN = 32) [ Required ];
    Property Password As %String(MAXLEN = 32);
    Property Name As %String(MAXLEN = 64);
}
```

2. tblEmployee (Shain Data)

```objectscript
Class SEM.tblEmployee Extends %Persistent
{
    Property EmployeeId As %String(MAXLEN = 5) [ Required ];
    Property Name As %String(MAXLEN = 64) [ Required ];
    // ... tsuika no properties
    Property RetireFlg As %Boolean [ Required ];
    Property deleteFlg As %Boolean [ Required ];
    Property upDateTime As %DateTime [ Required ];
}
```

Kono classes wa jidouteki ni IRIS de SQL tables wo sakusei shimasu. Kakko no instance wa global toshite stored sareru persistent object desu.

REST API Jissou:

REST API wa IRIS no %CSP.REST framework wo tsukatte defined sarete imasu:

```objectscript
```

```
Class SEM.SEMRESTAPI Extends %CSP.REST
{
  XData UrlMap
  {
    <Routes>
      <Route Url="/signup" Method="POST" Call="AccountRegistration"/>
      <Route Url="/signin" Method="POST" Call="AccountLogin"/>
      <Route Url="/employees" Method="GET" Call="GetAllEmployees"/>
      <Route Url="/employee/:id" Method="GET" Call="GetEmployeeById"/>
      <Route Url="/employee" Method="POST" Call="CreateEmployee"/>
      <Route Url="/employee/:id" Method="PUT" Call="UpdateEmployee"/>
      <Route Url="/employee/:id" Method="DELETE" Call="DeleteEmployee"/>
    </Routes>
  }
}
```

Kakko no route wa business logic wo handle suru ClassMethod ni map saremasu.

Shuyou na Backend Features:

1. **SQL Integration** - ObjectScript wa SQL queries wo chokusetsu jikkou dekiru
2. **Object Persistence** - Objects wa jidouteki ni database storage ni map sareru
3. **UTF-8 Support** - `$ZCONVERT()` function ga Nihongo characters wo handle suru
4. **Transaction Management** - ACID compliance wa built-in
5. **JSON Support** - Native JSON parsing to generation

Communication Flow

Frontend to backend ga dou communication suru ka setsumei shimasu:

Development Setup:

- Frontend wa Vite dev server de hashiru: `localhost:5173`
- Backend wa IRIS web server de hashiru: `localhost:52773`

CORS mondai wo sakeru tame, Vite no proxy configuration wo tsukaimasu:

```typescript
```

```
// vite.config.ts
export default defineConfig({
  server: {
    proxy: {
      '/sem': {
        target: 'http://localhost:52773',
        changeOrigin: true,
      },
    },
  },
});
```

Request Flow:

```
1. React Component ga API call wo suru: axios.post('/sem/employee', data)
2. Vite proxy ga forward suru: http://localhost:52773/sem/employee
3. IRIS web server ga request wo receive suru
4. SEMRESTAPI class ni route suru
5. CreateEmployee() method wo call suru
6. Business logic to database operations wo jikkou suru
7. JSON response wo return suru
8. Frontend ga response wo receive shite UI wo update suru
```

Data Format: Subete no communication wa JSON wo tsukaimasu. Tatoeba, shain wo sakusei suru toki:

Request:

```json
{
  "employeeId": "10001",
  "name": "山田花子",
  "kanaName": "ヤマダハナコ",
  "sex": 2,
  "department": "Engineering",
  "retireFlg": false
}
```

Response:

```json
```

```
{
  "message": "Employee created successfully",
  "status": "success"
}
```

Performance Considerations

Frontend:

- `useMemo` - takai computation no tame (filtering, sorting)
- Pagination - rendered rows wo limit suru
- Controlled components - re-renders wo minimize suru

Backend:

- Indexed database fields - hayai queries no tame
- SQL query optimization
- Kouka-teki na object loading - `%OpenId()` de

Security Considerations

Genzai no Jissou:

- Client-side form validation
- Server-side validation
- SQL query parameterization - injection wo prevent suru tame
- Soft delete - data hozon no tame

Production Enhancements:

- Password hashing (bcrypt)
- JWT token ninshou
- HTTPS angouka
- CORS configuration
- Rate limiting
- Input sanitization

🎯 Section 10: Technical Highlights (1 minute)

Shuyou na Gakushuu Points

Kono jissou no ikutsu ka no shuyou na gijutsu-teki aspects wo highlight shimasu:

1. React Hooks Mastery: Kono project wa React Hooks wo hirokute tsukatte imasu:

- `useState` - components wo tsuujite 15+ state variables no tame
- `useEffect` - data fetching to component lifecycle no tame
- `useMemo` - performance optimization no tame
- `useParams` to `useNavigate` - routing no tame

2. TypeScript no Merits: TypeScript wa compile-time type checking wo provide shimasu. Tatoeba:

```typescript
interface Employee {
  id?: number;
  EmployeeId: string;
  Name: string;
  // ...
}
```

Kore wa runtime errors wo prevent shi, code maintainability wo kaizen shimasu.

3. Material-UI Integration: Watashitachi wa Material-UI no comprehensive component library wo riyou shimasu:

- Form components (TextField, Select, RadioGroup)
- Data display (Table, Dialog)
- Navigation (AppBar, Toolbar)
- Feedback (Snackbar, CircularProgress)

4. ObjectScript no Nouryoku: Backend wa ObjectScript no power wo demonstration shimasu:

- Object-oriented programming
- Chokusetsu SQL integration
- Built-in JSON support
- UTF-8 character handling - `$ZCONVERT()` de
- DateTime functions - `$ZDATETIME()` to `$HOROLOG` de

5. IRIS Database Features:

- Persistent object storage
- ACID transactions
- Takai performance (in-memory globals)
- SQL compatibility
- Built-in web server

6. RESTful API Design: API wa REST principles ni shitagaimasu:

- Resource-based URLs (`/employee/:id`)
- HTTP methods wo semantically (GET, POST, PUT, DELETE)
- JSON data format
- Tadashii status codes

🚀 Section 11: Future Enhancements (0.5 minutes)

Kanousei no aru Kaizen

Shourai no kaihatsu no tame, ikutsu ka no enhancements ga kangaerareru:

Security Enhancements:

- Bcrypt de password hashing
- JWT token-based ninshou
- Role-based access control
- API rate limiting

Feature Tsuika:

- Advanced search - fukusuu no filters de
- Excel/PDF export kinou
- Bulk import/export
- Email notifications
- Audit log system
- Shain no shashin upload

UI/UX Kaizen:

- Dark mode support
- Internationalization (i18n) - fukusuu no gengo no tame
- Mobile-responsive design optimization
- Loading skeletons
- Advanced data visualization

Backend Kaizen:

- Caching layer (Redis)
- Comprehensive logging
- Jidou backup system
- Performance monitoring
- GraphQL API option

❓ Section 12: Q&A and Closing (3-4 minutes)

Shitsumon to Shumatsu

Kore de, Shain Kanri System no demonstration wo owarimasu.

Youyaku suru to, watashitachi wa cover shimashita:

1. Touroku kara shain kanri made no kanzen na user workflow
2. React, TypeScript, to Material-UI wo tsukatta frontend architecture
3. InterSystems IRIS to ObjectScript de backend jissou
4. Frontend to backend no aida no RESTful API communication
5. Soft delete, UTF-8 support, to state management wo fukumu shuyou na gijutsu-teki features

Ima, donna shitsumon demo ukemasu:

- Jissou no details
- Design decisions
- Chokumen shita gijutsu-teki kadai
- IRIS database features

- Tsukatta React.js patterns
- Matawa project no hoka no aspects

Gosei chou arigatou gozaimashita.

💡 Common Q&A Responses (Romaji)

Q: Naze hoka no frameworks yori React wo erabimashita ka? A: React wa component-based architecture to ookii ecosystem wo provide shimasu. Virtual DOM wa subarashii performance wo provide shi, hooks wa state management wo intuitive ni shimasu. Kuwaete, React wa tsuyoi TypeScript support to hiroi community resources ga arimasu.

Q: Naze JWT tokens no kawari ni localStorage desu ka? A: Kono demonstration project no tame, localStorage wa simplicity wo provide shimasu. Production kankyou de wa, yori yoi security no tame ni, HTTP-only cookies de JWT tokens, jidou expiration, to refresh token mechanisms wo jissou shimasu.

Q: IRIS wa dentou-teki na relational databases to dou chigaimasu ka? A: IRIS wa object to relational database capabilities no ryouhou wo provide shimasu. Objects wo globals toshite stores shi, in-memory speed wo provide shimasu. Built-in SQL support wa standard queries wo yuruushi, object model wa chokusetsu property access wo enable shimasu. Kono hybrid approach wa flexibility to performance wo provide shimasu.

Q: Naze hard delete no kawari ni soft delete desu ka? A: Soft delete wa audit trails to compliance requirements no tame ni data wo hozon shimasu. Machigatte deleted shita records no recovery wo yuruushi, referential integrity wo maintain shimasu. Kanzen na removal no tame, tekisetsu na authorization de betsu no purge function wo jissou dekimasu.

Q: Dou yatte concurrent updates wo handle shimasu ka? A: Genzai, system wa last-write-wins approach wo tsukaimasu. Production no tame, `upDateTime` field wo version indicator toshite tsukau optimistic locking, matawa IRIS no built-in transaction management wo pessimistic locking no tame ni jissou dekimasu.

Q: Scalability ni tsuite wa? A: Genzai no architecture wa vertical scaling wo yoku support shimasu. Horizontal scaling no tame:

- Web tier no tame no load balancer wo jissou suru
- IRIS no distributed cache capabilities wo tsukau
- Read to write operations wo bunri suru
- Redis no youna caching layer wo tsuika suru

Q: Nihongo text encoding wa dou handle saremasu ka? A: Zentai de UTF-8 encoding wo tsukaimasu. Frontend de, JavaScript wa natively UTF-8 wo support shimasu. Backend de, ObjectScript no `$ZCONVERT(text, "I", "UTF8")` function wa database kara Nihongo text wo store to retrieve suru toki ni tadashii character encoding wo ensure shimasu.

Q: Ninshou flow wo kuwashiku setsumei dekimasu ka? A:

1. User ga credentials wo submit suru
2. Frontend ga POST wo `/sem/signin` ni okuru
3. Backend ga `tblAccount` table wo query suru
4. Password hikaku (demo de wa plain text, production de wa hashed)
5. Seikou sureba, frontend ga `isLoggedIn` to `userEmail` wo localStorage ni hozon suru
6. `ProtectedRoute` component ga protected pages wo render suru mae ni localStorage wo check suru
7. Logout ga localStorage wo clear shite sign-in ni redirect suru

---

[End of Presentation]

Gochuumon arigatou gozaimashita. Kono kikai wo itadaki, kansha itashimasu.

[Ojigi wo suru]