**TOGAF Architecture Document: Traveller Character Sheet Application**

**1. Executive Summary**

The Traveller Character Sheet is a client-side web application designed to support players of the Mongoose Traveller 2nd Edition role-playing game. It enables character creation, management, storage, and printing in a digital format. This architecture document describes the design, components, and technical decisions supporting this application according to The Open Group Architecture Framework (TOGAF).

This document is intended for developers, maintainers, and stakeholders interested in the technical architecture of the application. It serves as both documentation and a guide for future development.

**2. Architecture Vision**

**2.1 Problem Statement**

Players of the Traveller role-playing game need a digital tool to create and manage character sheets, replacing traditional paper sheets with a solution that adds automation, validation, and persistence.

**2.2 Vision Statement**

To provide the most comprehensive and user-friendly digital character sheet for Traveller 2nd Edition that works entirely within a web browser, requires no server connection after initial load, and supports complete character lifecycle management.

**2.3 Stakeholders**

* **Primary Users**: Players and Game Masters of the Traveller RPG
* **Developers**: Application maintainers
* **Content Providers**: Mongoose Publishing (indirectly, as IP holders)

**2.4 Key Drivers**

1. **Accessibility**: Available to any user with a web browser
2. **Offline Capability**: Functional without ongoing internet connection
3. **Comprehensiveness**: Support for all character creation and management rules
4. **User Experience**: Intuitive interface requiring minimal learning curve
5. **Portability**: Usable across different devices and platforms

**2.5 Constraints**

1. **Legal**: Must respect Traveller intellectual property
2. **Technical**: Must operate without server-side processing
3. **Storage**: Limited to browser local storage capacity
4. **Browser Compatibility**: Must support modern browsers only

**3. Business Architecture**

**3.1 Business Goals and Objectives**

1. Provide a complete digital alternative to paper character sheets
2. Automate complex character calculations and validations
3. Enable persistent character data across sessions
4. Support import/export for character sharing
5. Facilitate printing for tabletop use

**3.2 Business Processes**

1. **Character Creation Process**
   * Input basic character information
   * Establish characteristics
   * Select pre-career education
   * Choose and record careers
   * Acquire skills and equipment
   * Calculate derived attributes
2. **Character Advancement Process**
   * Add new skills or increase skill levels
   * Track skills in training
   * Update characteristics as they change
   * Add new career entries
   * Update equipment and finances
3. **Game Session Support Process**
   * Roll dice for task resolution
   * Record changes to character state
   * Track resources and finances
   * Print character information

**3.3 Business Capabilities**

1. Character data management
2. Rules automation
3. Data persistence
4. Document generation (printing)
5. Dice simulation

**3.4 Organization Structure**

As a client-side web application, there is no formal organizational structure. Development follows an open-source model with contributions managed through version control.

**4. Information Systems Architecture**

**4.1 Data Architecture**

**4.1.1 Logical Data Entities**

1. **Character**: Core entity representing the player character
2. **Characteristics**: Physical and mental attributes (STR, DEX, END, INT, EDU, SOC, etc.)
3. **Skills**: Acquired abilities with proficiency levels
4. **Training Skills**: Skills currently being learned
5. **Education**: Pre-career educational experiences
6. **Career**: Professional history entries
7. **Weapons**: Combat equipment
8. **Armor**: Protective equipment
9. **General Equipment**: Other possessions
10. **Finances**: Money, debt, and ongoing expenses

**4.1.2 Data Model**

Character

├── Basic Information (name, species, age, etc.)

├── Characteristics (current and baseline values)

├── Skills (with specializations and levels)

├── Skills in Training (with weeks spent)

├── Education History (types, years, outcomes)

├── Career History (careers, assignments, ranks)

├── Equipment

│ ├── Weapons

│ ├── Armor

│ └── General Equipment

└── Finances (credits, pension, debt, etc.)

**4.1.3 Data Storage**

Primary storage mechanism is browser localStorage, with JSON as the data format. Additional import/export capability uses JSON file download/upload.

**4.2 Application Architecture**

**4.2.1 Application Components**

1. **User Interface Components**
   * Form elements for data input
   * Dynamic containers for repeating elements
   * Print layout formatting
2. **Business Logic Components**
   * Characteristic calculation functions
   * Skill management functions
   * Career and education tracking
   * Age and experience calculation
   * Dice rolling simulation
3. **Data Management Components**
   * LocalStorage interaction
   * Export/import functions
   * Character state management

**4.2.2 Application Interactions**

1. **UI to Business Logic**: Event listeners capture user input and invoke processing functions
2. **Business Logic to Data Management**: Processing functions update character state in memory
3. **Data Management to Storage**: Save functions persist data to localStorage
4. **Storage to Data Management**: Load functions retrieve data from localStorage
5. **Data Management to UI**: Retrieved data is rendered into UI components

**4.2.3 User Interface Architecture**

The UI follows a sectioned layout with collapsible panels organized by character information type:

* Header and identification
* Characteristics
* Additional characteristics
* Skills in training
* Skills
* Personal data
* Pre-career options
* Career history
* Weapons & attacks
* Armor
* Augments
* Equipment
* Finances
* Notes
* Utility functions (dice roller, save/load, print)

**5. Technology Architecture**

**5.1 Software Components**

1. **Front-end Framework**: HTML5, CSS3, and JavaScript (ES6+)
2. **CSS Framework**: Bootstrap 5.3 for responsive design
3. **Typography**: Google Fonts (Arsenal, Marcellus, Roboto, Arimo, Work Sans)
4. **Storage API**: Browser localStorage
5. **File API**: Browser File System Access

**5.2 Physical Architecture**

Client-side only application running in web browsers with no server-side components.

**5.3 Technology Standards**

1. HTML5 for structure
2. CSS3 for styling
3. ECMAScript 2015+ for functionality
4. JSON for data serialization
5. Web Storage API for persistence

**5.4 Security Architecture**

1. Data remains local to the user's device
2. No authentication required
3. No sensitive data processed or transmitted

**6. Implementation and Migration Planning**

**6.1 Implementation Strategy**

This is a greenfield development with no legacy system to migrate from.

**6.2 Development Approach**

1. Initial development as a static HTML/JS/CSS application
2. Feature enhancement through iterative development
3. Modularization of JavaScript components

**6.3 Implementation Sequence**

1. Core character data structure and UI layout
2. Basic characteristic and skill management
3. Career and education tracking
4. Equipment and finance tracking
5. Data persistence (save/load)
6. Import/export functionality
7. Print optimization

**7. Architecture Governance**

**7.1 Development Standards**

1. JavaScript code follows ES6+ standards
2. HTML5 semantic markup
3. CSS using Bootstrap conventions with custom extensions
4. Consistent naming conventions across code

**7.2 Testing Approach**

1. Manual testing across supported browsers
2. Validation of calculations against game rules
3. User acceptance testing with Traveller players

**7.3 Maintenance Process**

1. Issue tracking through GitHub
2. Version control with Git
3. Documentation updates with code changes

**8. Change Management**

**8.1 Version Control Strategy**

1. Git for source code management
2. Feature branches for development
3. Main branch for stable releases

**8.2 Release Management**

1. Semantic versioning (MAJOR.MINOR.PATCH)
2. GitHub releases for distribution
3. Change log documentation

**8.3 Future Enhancements**

1. Service worker for improved offline capabilities
2. Character sharing via cloud storage options
3. Integration with virtual tabletop platforms
4. Support for Traveller expansions and supplements
5. Character creation wizard
6. Advanced dice roller with visual feedback

**9. Appendices**

**9.1 Technology Stack Details**

* **HTML5**: Structure and semantic markup
* **CSS3/Bootstrap 5.3**: Styling and responsive design
* **JavaScript (ES6+)**: Application logic
* **LocalStorage API**: Data persistence
* **File API**: Import/export functionality

**9.2 Data Dictionary**

|  |  |  |
| --- | --- | --- |
| Entity | Description | Key Attributes |
| Character | Core player character data | name, species, age, homeworld |
| Characteristic | Physical or mental attribute | type, current, baseline, DM |
| Skill | Learned ability | name, specialization, level |
| TrainingSkill | Skill being learned | skill, specialization, weeksSpent |
| Education | Pre-career education | type, years, outcome, benefits |
| Career | Professional history | career, assignment, years, rank, benefits |
| Weapon | Combat equipment | name, TL, damage, range, weight, magazine |
| Armor | Protective equipment | name, rating, TL, radiation |
| Equipment | General possession | name, TL, mass, cost |
| Finances | Money and expenses | credits, pension, debt, cashOnHand |

**9.3 Architecture Decisions**

**AD01: Client-Side Only Architecture**

**Decision**: Implement as a client-side only application with no server components.  
**Rationale**: Maximizes accessibility, eliminates hosting costs, and allows offline usage.  
**Implications**: Storage limited to browser capabilities; no multi-user or cloud synchronization.

**AD02: Local Storage for Persistence**

**Decision**: Use browser localStorage for data persistence.  
**Rationale**: Available in all modern browsers, requires no authentication, works offline.  
**Implications**: Limited storage space (typically 5-10MB); data confined to specific browser/device.

**AD03: Bootstrap for Responsive UI**

**Decision**: Use Bootstrap 5.3 as the CSS framework.  
**Rationale**: Provides responsive design patterns, cross-browser compatibility, and familiar UI components.  
**Implications**: Additional dependency; some styling limitations requiring custom CSS.

**AD04: JSON for Data Exchange**

**Decision**: Use JSON for all data serialization needs.  
**Rationale**: Native JavaScript support, human-readable, compact representation.  
**Implications**: Cannot store functions or complex object types directly.