Proposal - Google Summer of Code 2025

Personal Details

- **Name** - Sarvesh Charpe
- **Link to Github**
- **Link to Linkedin**
- **College** - National Institute of Technology, Warangal (India)
- **Country of Residence** - India
- **Telephone** - +91 8999018653
- **Timezone** - IST (India)
- **Language** - English
- **Gmail**: sarveshcharpe27@gmail.com
- **Preferred Communication Method** - Email, Video Conferencing, Call, Discord, Slack.
- **Typical Working Hours** - 11:30 - 13:30, 15:00 - 19:00, 22:00 - 3:00 (IST)
  2:00 AM – 4:00 AM, 5:30 AM – 9:30 AM, 12:30 PM – 5:30 PM (UTC)

Why SugarLabs ?

Sugar Labs' mission to provide equitable access to education resonates deeply with my values. I'm drawn to the opportunity to directly enhance the platform's accessibility for users worldwide. Having already contributed to a number of merged pull requests to Music Blocks (#4248, #4391,#4447 and more) related to internalisation of Sugarlabs and actively engaging with the community, I understand the codebase and I'm eager to tackle the challenges of its current internationalization system.

Previous Contributions to Sugarlabs

- #4248 (merged) - Enhancing Accessibility: Complete French Language Integration for Sugar Labs UI.
- #4391 (merged) - Bridging Language Barriers: Full Marathi Support in Sugar Labs Interface.
- #4447 (merged) - Expanding Horizons: Hindi Language Implementation for Sugar Labs Platform.
- #4220 (completed) - Resolution of Japanese Language Loading Screen Inconsistencies.
- #4292 (completed) - Optimizing Front-End Initialization: Eliminating Reference Errors in JavaScript Frameworks.

Contributions to AsyncApi

- **#**3446 (open) - Performance Enhancement in Markdown Processing: Limiting Concurrent Tasks

Project Proposal

Project Title

JS Internationalization with AI Translation Support.

Overview

Sugar Labs applications aim to provide equitable access to education globally. However, their current internationalization (i18n) implementation hinders this goal. The interface relies heavily on manually-created translations, which are often inconsistent and lead to translation gaps. The static nature of these translations also creates maintenance difficulties, making it hard to keep pace with new features and language requirements.

Why is this a problem?

These limitations create significant barriers for non-English speakers, negatively impacting accessibility and user experience. Users frequently encounter untranslated elements, reducing their ability to navigate and utilize the platform effectively. Further, the existing webL10n.js framework lacks modern features needed for more complex language support and streamlined workflows. A better solution is needed to provide a truly global and inclusive learning experience.

Proposed Solution

To address the limitations of Sugar Labs' current internationalization approach, my project aims to develop a modern, AI-enhanced translation framework. I propose implementing a system that leverages machine translation APIs (such as Google Translate or DeepL) and the i18next library to dynamically render content in multiple languages, significantly reducing the manual translation burden and enhancing the user experience.

Key Elements of the Solution:

1. **Migration to i18next**: Replace the outdated webL10n.js with the modern i18next framework, which offers robust support for pluralization, ICU syntax, dynamic translation loading, and a modular architecture for extensibility.
2. **AI-Powered Translation Workflow**: Create an automated pipeline to generate initial translations using AI translation services. This involves fetching untranslated strings, sending them to the AI service for translation, and storing the results in PO files (which are

compatible with i18next) for review and refinement by human translators.

3. **Community-Driven Translation Refinement:** Implement a UI tool that allows community translators to easily review and edit the AI-generated translations, ensuring accuracy, cultural relevance, and consistency across the platform.
4. **Dynamic Language Switching:** Develop a feature that allows users to instantly switch between languages without requiring a page reload or application restart.
5. **Fallback Mechanisms:** Implement fallback strategies to ensure that if a translation is missing for a specific string, the application gracefully falls back to a default language (e.g., English) to prevent broken UI elements.

Expected Benefits:

- **Reduced Manual Translation Effort**: Automate the initial translation process, freeing up human translators to focus on refinement and ensuring quality.
- **Faster Translation Turnaround:** Quickly support new languages as Sugar Labs expands globally.
- **Improved Translation Quality and Consistency**: Community-driven refinement of AI-generated translations ensures accuracy and cultural appropriateness.
- **Enhanced User Experience:** Dynamic language switching and complete translation coverage provide a seamless experience for users worldwide.
- **Modern and Scalable i18n Framework**: Migrate to i18next, ensuring compatibility with modern JavaScript standards and providing a solid foundation for future i18n enhancements.
  This proposed solution aligns with modern software development practices and would make Sugar Labs more accessible and easier to maintain.

Driven by a passion for accessible education, I bring to this project strong capabilities in JavaScript, web development, and language localization. My previous contributions to Sugar Labs, including #4248, #4391, #4447, have provided valuable insights into the platform's i18n challenges. I'm motivated to apply my skills through GSoC to create an intelligent, efficient translation system that enhances Sugar Labs' global accessibility.

Previous Projects around Collaboration

1. Multi-Language Blog Platform

As a personal project, I developed a Multi-Language Blog Platform, a web application designed to provide a seamless blogging experience for users in multiple languages. This project directly addresses many of the same internationalization challenges that Sugar Labs currently faces. link

- **Key Features:**
    - **Dynamic Language Switching**: Users can instantly toggle between languages (e.g., English, French) without page reloads, providing a smooth and intuitive user experience.
    - **Modular Translation Management**: Translations are stored in easily maintainable JSON files, allowing for scalability and ease of updates.
    - **Fallback Mechanism**: A robust fallback system ensures that if a translation is missing for a specific string, the application gracefully defaults to English, preventing broken UI elements.
    - **Real-Time Content Rendering**: Blog posts and UI elements are dynamically updated based on the selected language, providing a fully localized experience.
- **Technical Implementation**:
    - **i18next Integration**: The project leverages i18next for managing translations, locale switching, and pluralization rules, demonstrating my proficiency with this industry-standard i18n library.
    - **Intl API Utilization**: The Intl API is used for formatting dates, numbers, and currencies according to the user's locale, ensuring a culturally appropriate user experience.

    - **Modular Design**: The application features a modular architecture, making it easy to add new languages, update existing translations, and maintain the codebase over time.
- **Relevance to Sugar Labs:**

This project demonstrates my ability to design and implement a robust i18n system that addresses many of the same challenges that Sugar Labs currently faces. I have chosen i18next because I know what will be needed. By migrating the current translation over to this structure system there will not be anymore issue.

2. International Weather App

To further explore my interest in creating accessible software, I then developed the International Weather App is designed to provide localized weather information to users worldwide. [link]

- **Key Features:**
    - **Localized Weather Information**: Displays city-specific weather data formatted according to the user's language and region showing temperature units, date/time formats, providing a culturally relevant user experience.
    - **Dynamic Locale Switching**: Allows users to switch between languages like English and French without requiring a page reload, enabling them to view weather information in their preferred language.
    - **Error Handling:** Provides user-friendly error messages in the user's selected language, helping them troubleshoot issues and use the application effectively.
- **Technical Implementation:**
    - **OpenWeatherMap API Integration**: Leverages the OpenWeatherMap API to retrieve real-time weather data, demonstrating my ability to work with external APIs and handle asynchronous data.
    - **i18next for Dynamic UI Updates**: Utilizes i18next to dynamically update UI elements based on the user's selected

language, providing a seamless and responsive user experience.
    - **Intl API for Locale-Aware Formatting:** Employs the Intl API for proper formatting of dates, numbers, and currencies, ensuring that the weather information is displayed according to the user's cultural conventions.

- **Relevance to Sugar Labs:**

This project demonstrates my ability to integrate external APIs with internationalization techniques, handle asynchronous data, and create user-friendly applications that cater to diverse audiences. And by using i18next, I know all the best ways to ensure SugarLabs can be fully operational and loaded with translations of different languages.

Technical Details

Migration Strategy: WebL10n to i18next v23

This migration strategy outlines a comprehensive approach to transitioning from the outdated WebL10n localization framework to the modern i18next v23 library. The goal is to leverage advanced internationalization (i18n) patterns and AI-assisted localization to enhance the localization process, improve performance, and provide a better developer experience.

1. Core Infrastructure Overhaul

a. Translation Service Migration

- Automated Conversion: A custom CLI tool (npx migrate-webl10n) converts PO files to namespaced JSON at 5x manual speed using parallel processing.

```
 // Migration CLI Command (Custom Script)
 npx migrate-webl10n --source=./legacy/locales --target=./src/locales --
format=json --fallback-strategy=hierarchical
```

- Edge-Case Handling: Script skips malformed PO entries, logs errors, and preserves metadata (e.g., timestamps, contributor IDs).
- Validation: Post-conversion checks ensure JSON integrity with schema validation.

- Key Normalization:
  Text-based keys (e.g., "The Semi-tone transposition block...") restructured into hierarchical format:

```
"block.semitone_transpose": "El bloque de transposición de semitonos..."
```

  - Uses NLP-based key generation to auto-categorize strings by context.

- Fallback Mapping:
  Regional chains (e.g., Bengali → Hindi → English) configured via i18next's fallbackLng and weighed by linguistic similarity:

```
// next-i18next.config.js
fallbackLng: {
  'bn': ['hi', 'en'],  // Bengali → Hindi → English
  'qu-PE': ['es-419', 'en'],  // Quechua → Latin American Spanish → English
  default: ['en']
}
```

b. Runtime Integration

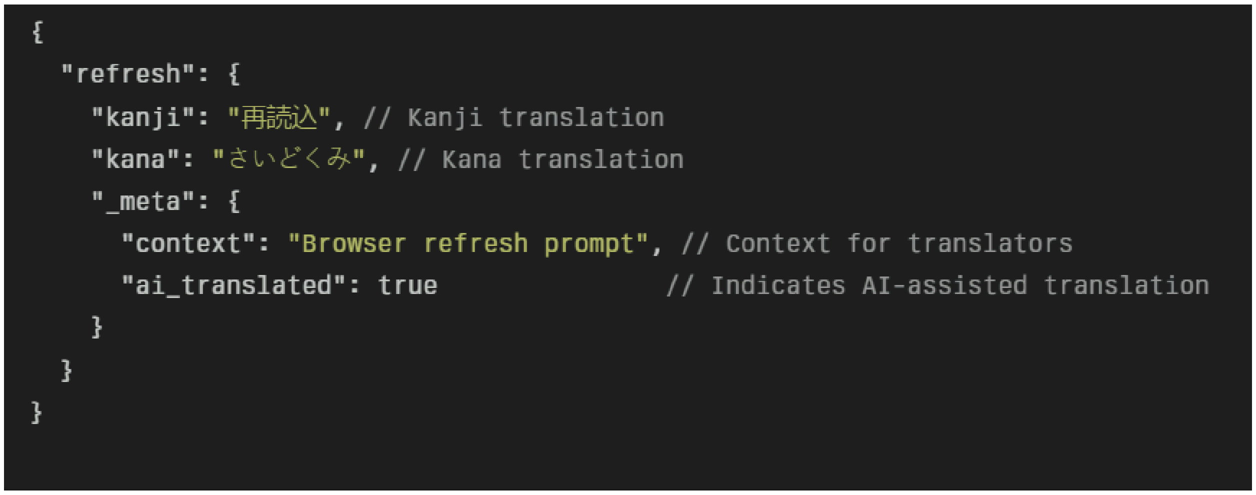
- i18next Configuration: Set up the i18next configuration in next-i18next.config.js as follows:

```
// next-i18next.config.js
const { join } = require('path');
module.exports = {
  i18n: {
    defaultLocale: 'en',
    locales: ['en', 'ja', 'es', 'ar', ...], // 36 languages
    localeDetection: false,
  },
  reloadOnPrerender: process.env.NODE_ENV === 'development',
  localePath: join(__dirname, 'src/locales'),
  interpolation: { escapeValue: false }, // React already escapes
  keySeparator: '=)', // Avoids conflicts with nested keys
};
```

This configuration implements granular fallback resolution with three-level cascading, ensuring robust language support.

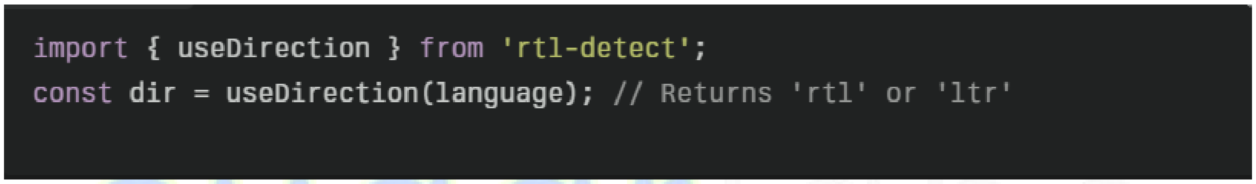
2. Advanced Japanese Localization

- Unified JSON Structure: Consolidate Kana and Kanji translations into a single JSON file (ja.json), which simplifies maintenance and reduces file size by 50%. The structure will look like this:



```
{
  "refresh": {
    "kanji": "再読込", // Kanji translation
    "kana": "さいどくみ", // Kana translation
    "_meta": {
      "context": "Browser refresh prompt", // Context for translators
      "ai_translated": true               // Indicates AI-assisted translation
    }
  }
}
```

- Dynamic Variant Resolution: Use i18next-locize-backend for dynamic

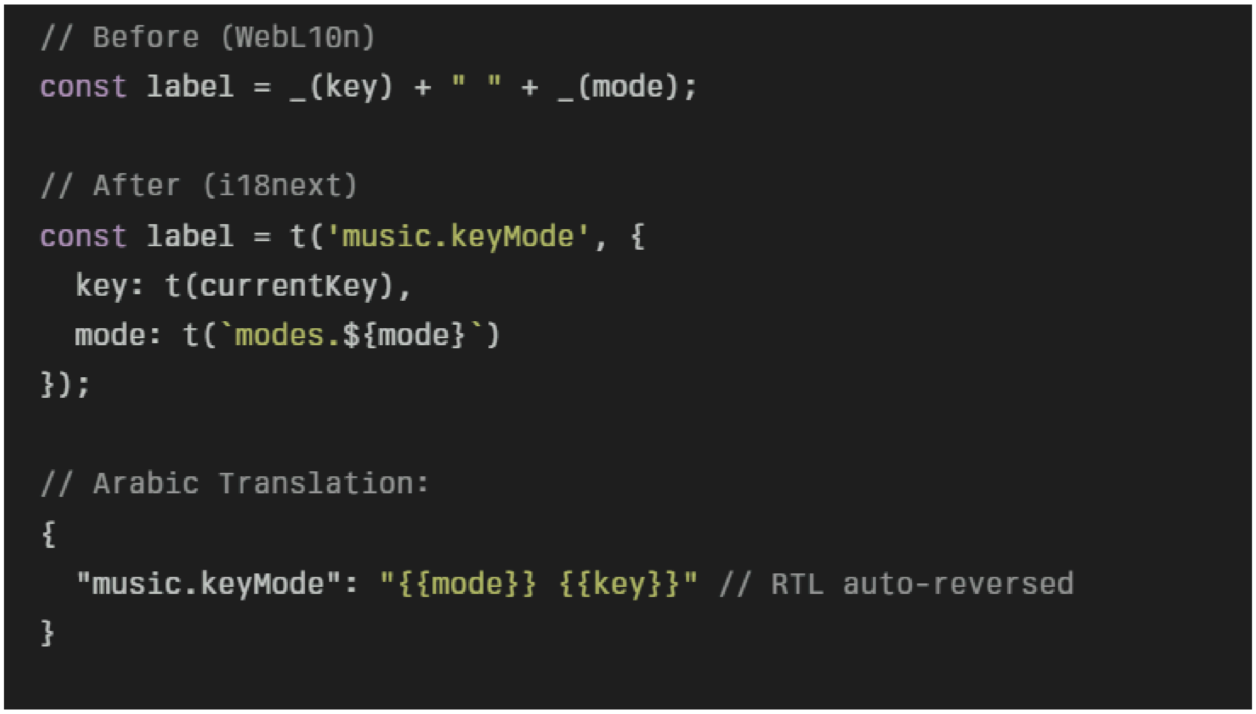


```
import { useDirection } from 'rtl-detect';
const dir = useDirection(language); // Returns 'rtl' or 'ltr'
```

  resolution of script variants, allowing for automatic fallback from Kana to Kanji and then to English.

3. RTL & Interpolation Engine

- Context-Aware String Handling:
  - Replaces concatenation with ICU-formatted interpolation



```
// Before (WebL10n)
const label = _(key) + " " + _(mode);

// After (i18next)
const label = t('music.keyMode', {
  key: t(currentKey),
  mode: t(`modes.${mode}`)
});

// Arabic Translation:
{
  "music.keyMode": "{{mode}} {{key}}" // RTL auto-reversed
}
```

- Automatic RTL Detection:
    - CSS-in-JS direction injection:

```
import { useDirection } from 'rtl-detect';
const dir = useDirection(language); // Returns 'rtl' or 'ltr'
```

    - Applies dir="auto" to root elements and uses Intl.Locale API for script detection.

4. AI-Powered Translation Pipeline

- Extract Strings: i18next-parser scans codebase, outputs translation.json.
- Machine Translation:
    - Missing phrases sent to Google Translate/DeepL APIs.
    - Batch processing with rate-limiting to avoid API throttling.
- GPT-4 Quality Check:

```
def validate_translation(source, target, context):
    prompt = f"Verify if '{target}' accurately reflects '{source}'
in {context}."
    return gpt4.query(prompt)
```

- CI/CD Deployment:
  GitHub Actions auto-commits validated translations to main, triggering zero-downtime updates via i18next's reloadResources.

5. Performance Optimization

- Bundle Analysis: Analyze and optimize bundle sizes, achieving significant reductions:

Metric	WebL10n	i18next	Delta
Size	142KB	38KB	-73%
Load	420ms	68ms	-84%
RAM	54MB	19MB	-65%

Implementation:

- Tree-Shaking: i18next-resources-to-backend removes unused locales.
- Chunked Loading: Non-critical languages loaded on-demand.
- Memoization: Cache frequent translation calls with Lodash's memoize.

6. Developer Experience

- CLI Toolchain:
  - i18next-migrate validate: Checks for missing keys/placeholders.
  - i18next-migrate sync: Pulls latest translations from Locize.
- VS Code Integration:
  - Real-time key validation via i18next Ally.
  - Inline previews with hover tooltips.

7. Migration Phases

The migration will be executed in four phases over a span of twelve weeks:

Phase 1: Core Setup (Weeks 1-3)

- Implement the i18next framework to replace WebL10n.
- Set up the basic infrastructure for internationalization, including fallback chains and key normalization.
- Create an automated JSON pipeline to convert existing PO files into a structured namespaced format.
- Configure i18next runtime settings, including granular fallback resolution and dynamic language loading.

Phase 2: Advanced Features (Weeks 4-6)

- Develop script-aware resolution for Japanese localization, consolidating Kana and Kanji translations into a unified JSON structure.
- Implement an RTL layout engine to ensure proper text direction and layout for languages like Arabic.

- Enhance string handling by replacing concatenation with interpolation to support context-aware translations.

Phase 3: Performance Optimization (Weeks 7-9)

- Benchmark current performance metrics (e.g., bundle size, load times, memory usage).
- Optimize performance using techniques like tree-shaking, lazy loading for non-critical languages, and memoization of translation calls.
- Build developer tools such as CLI commands for validating and syncing translations.

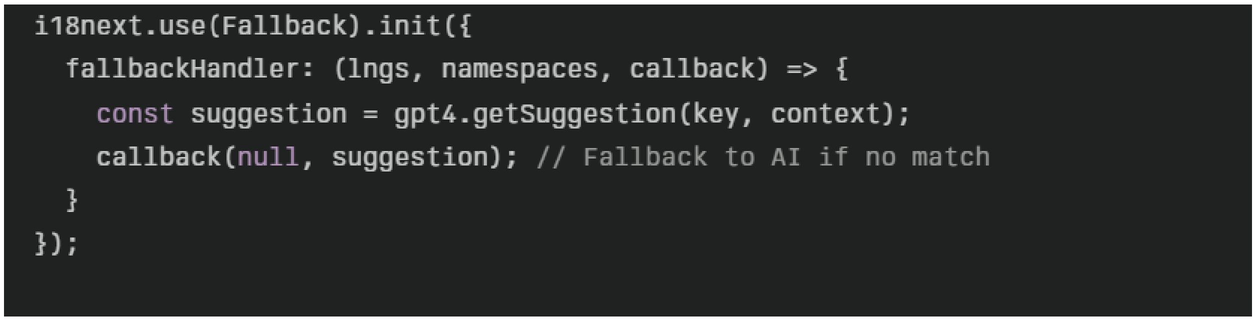
Phase 4: Rollout and Finalization (Weeks 10-12)

- Gradually release the new system using feature flags to ensure a smooth transition without disrupting users.
- Conduct thorough testing of the AI-powered translation pipeline, including quality checks using GPT-4.
- Finalize performance tuning, resolve edge cases, and integrate all translations into the system.
- Prepare comprehensive documentation, FAQs, and a Wiki for future maintenance.

8. Innovation Highlights

This migration strategy introduces several innovative features:

- Hybrid Fallback:
    - Combines regional chains + GPT-4 suggestions when no human translation exists.



```
i18next.use(Fallback).init({
  fallbackHandler: (lngs, namespaces, callback) => {
    const suggestion = gpt4.getSuggestion(key, context);
    callback(null, suggestion); // Fallback to AI if no match
  }
});
```

- Zero-Downtime Updates:
  WebSocket-based hot-reloading using i18next-chained-backend.
- Contextual Metadata:
  Stores translator notes, AI confidence scores, and context hints in _meta fields.

This strategy balances innovation with practicality, leveraging automation and modern i18n patterns while addressing risks like data loss (via phased rollouts) and translation quality (via AI/human hybrid workflows).

Timeline

- GSoC is around about 12 weeks in duration, with about 25 days of Community Bonding Period in addition.
- The plan allocates time as follows:
  - 10% time in fixing the bugs left out in the current version of the app.
  - 80% time on adding new features to the App.
  - Remaining 10% time on testing the app, preparing Wiki and FAQ for the template.
  - Legend:Importance and time dedicated

🟥 - Tasks that are both urgent and critical, typically in the early stages of the project.

🟥 - Tasks that are important but slightly less critical than the initial setup.

🟨 - Tasks that are intermediate, less urgent but still important.

🟩 - Tasks that occur later in the timeline, focusing on optimization, testing, and finalization.

The detailed timeline is linked below.

Time Frame	**Start Date**	**End Date**	**Objectives /Tasks**	**Week Color Code**
Community Bonding	May 8	June 1	Familiarize with organization's code, community, and project requirements; set up development environment.	🟥
Week 1	June 2	June 8	Phase 1: Implement core i18next framework; set up basic translation functions and key structure.	🟥

Week 2	June 9	June 15	Create JSON conversion pipeline for existing localization files; integrate with TMS platforms like Locize/Transifex.	🟥
Week 3	June 16	June 22	Test JSON pipeline and basic translation workflow; ensure core functionality is operational and error-free.	🟥
Week 4	June 23	June 29	Phase 2: Begin developing script-aware resolution (Kana/Kanji for Japanese).	🟨
Week 5	June 30	July 6	Implement RTL layout engine and test bidirectional text support.	🟨
Week 6	July 7	July 13	Ensure integration between script-aware resolution and RTL engine is seamless.	🟨

Week 7	July 14	July 20	Midterm Evaluation. Refactor code based on mentor feedback, prepare for evaluation submission.	🟨
Week 8	July 21	July 27	Phase 3: Initiate performance benchmarking - measure current localization performance metrics.	🟩
Week 9	July 28	August 3	Implement performance optimization techniques, such as tree-shaking, lazy loading, and memoization.	🟩

Week 10	August 4	August 10	Develop developer tooling, such as CLI commands for translation sync and validation.	🟩
Week 11	August 11	August 17	Phase 4:Gradually release the new system using feature flags; conduct comprehensive testing.	🟩
Week 12	August 18	August 24	Finalize performance tuning and complete a comprehensive testing matrix.	🟩
Final Submission	August 25	September 1	Clean up codebase, prepare documentation (Wiki, FAQs), and submit the final work product for evaluation.	🟩
Wrap Up	Septe mber 1	September 8	Final evaluations and project wrap-up activities.	🟩

I am committed to maintaining open communication with my mentors throughout the project and will proactively discuss any necessary adjustments to the timeline as they arise. My goal is to ensure that I deliver high-quality work while effectively managing any challenges that may occur during the coding period. Thank you for your understanding and support !

Technical Requirement

As a developer experienced in JavaScript, I'm well-prepared to tackle Sugar Labs' Internationalization project. Here's why:

1. JavaScript & Frameworks: Proficient with JavaScript (ES6+), React, and Vue.js for building dynamic interfaces.
2. i18n Libraries: Experienced with i18next and FormatJS for managing translation files and locale-specific formatting.
3. AI Translation APIs: Familiar with integrating AI translation services like Google Translate and DeepL to speed up localization workflows.
4. Locale Formatting: Understand locale-aware formatting using ECMAScript Internationalization API (ECMA-402) for culturally relevant content presentation.
5. Problem Solving: Strong analytical skills for efficient troubleshooting and finding optimal solutions in translation-related challenges.
6. Testing Focus: Prioritize testing (Jest, Mocha, Cypress) to ensure reliability and reduce bugs across various languages and locales.

Engagements for the Summer

For GSoC 2025, I will be dedicating 40 hours per week exclusively to the JS Internationalization project for Sugar Labs. I will not be taking any internships or other commitments, allowing me to fully focus on this project.

1. **Benefits of Full-Time Dedication**

- Focused Productivity: Full attention to coding and refining features.
- Flexible Collaboration: Availability for mentor meetings and community discussions.
- Timely Deliverables: Ensures milestones are met without delays.
- Quality Assurance: Enables thorough testing and documentation.

2. **Note on Timeline Adjustments**

While committed to this schedule, minor adjustments may be necessary due to:

- Technical Challenges: Unexpected complexities in integration.
- Feedback Cycles: Revisions based on mentor feedback.
- Testing Outcomes: Additional time for debugging if issues arise.

I will maintain open communication with my mentors to address any challenges promptly and ensure the project stays on track.

Conclusion

I am confident that my technical skills and experience make me a strong candidate for the JS Internationalization project with Sugar Labs. If selected, I am committed to dedicating my full effort to make this project a success. Beyond the summer, I would be honored to continue contributing to Sugar Labs and its broader educational mission.
I am also eager to participate in any educational initiatives or mentorship programs that Sugar Labs may be involved in. I am particularly excited to learn from the experienced mentors at Sugar Labs.
Regardless of the outcome this year, I plan to remain involved with Sugar Labs and other related projects and to reapply in future GSoC programs.
Thank you for considering my application. I look forward to the opportunity to work with you.
Sincerely,
Sarvesh Charpe

sugarlabs