Google Summer of Code 2025 Proposal

Contact Information

Sebastian Vivas
Email: seb.vivas.dev@gmail.com
LinkedIn: linkedin.com/in/sebvivas
GitHub: github.com/sebastian-vivas

Project Title

Block Search and Navigation System for MIT App Inventor

Project Size

Large (350 hours)

Project Technologies

Java, JavaScript, GWT, UIBinder, Search Algorithms, DOM Manipulation, Data Indexing

Project Topics

Code Navigation, Search Systems, Educational Technology, User Experience, Developer Tools

Proposal Summary

App Inventor projects can quickly grow complex, with hundreds of blocks spread across multiple procedures, event handlers, and screens. However, users have no efficient way to search through their blocks or locate specific components within their code, making project navigation and maintenance difficult as projects scale.

I propose to develop a comprehensive Block Search and Navigation System that enables users to quickly locate blocks and components within their App Inventor projects. This system will make large projects more manageable and help users navigate complex codebases efficiently.

This search system will greatly improve the usability of App Inventor for complex projects by giving users powerful tools to navigate and maintain their code, which will address a pain point for experienced users who work with large block-based programs.

The implementation will use Java/GWT for backend integration with App Inventor's core architecture and JavaScript for the interactive search interface, ensuring a seamless experience within the existing platform while maintaining performance even with large projects.

Benefits to Community

I found several forum posts from App Inventor users struggling with the lack of search functionality. Users currently export their code to text editors just to find where components are used. My solution directly addresses this problem and benefits the community in key ways:

Educational settings will see immediate improvements. Teachers and students will spend less time searching through code and more time focusing on learning programming concepts. Complex classroom projects will become more manageable.

Advanced users can build and maintain larger applications without switching to text-based environments. This extends App Inventor's usefulness beyond small projects and increases its adoption for more serious development work.

The open source community gains a feature that demonstrates block-based environments can support complex projects. This positions App Inventor as a more competitive option against traditional IDEs for certain use cases.

Problem Statement

I looked through App Inventor's Help threads to gain insight on what users need. I found this post, where a user described their struggle:

> "I am wondering if there is any way to search large screens full of blocks for mentions of a certain component. I am using a few clocks for triggering events. I have a large amount of procedures which I group by function type. (P11 is for procedures accessing database for get) (P12 is for procedures accessing database for set) I keep my functions closed to keep my sanity, I wonder if there is a way to find blocks that use a component."

Users currently resort to copying blocks into text editors to search for components. App Inventor lacks a built-in solution for finding where specific components are used, especially when procedures are collapsed to manage complexity.

Solution Approach

I will implement the Block Search and Navigation System as an integrated feature within the App Inventor Blocks Editor. The search interface will help users find blocks by:

- Component name or type
- Text content within blocks
- Block type or category

The system will work with collapsed procedures and provide clear visual indicators for search results with easy navigation between findings.

My technical approach includes:

- Creating an indexing system for all blocks in the workspace
- Implementing efficient search algorithms that traverse complex block structures
- Building a user-friendly search interface integrated into the existing UI
- Ensuring good performance even with large projects

Deliverables

MVP Deliverables:

1. **Integrated search interface in the Blocks Editor** to provide basic search functionality that needs to be integrated into the UI.
2. **Component-based filtering** to directly address the user's primary need to search for blocks that use specific components like clocks.
3. **Text-based search across block labels and components** because basic text search capability is fundamental.
4. **Result highlighting and navigation system** to jump to found blocks, especially within collapsed procedures.
5. **Performance optimization for large projects** because the system must work efficiently even with complex projects.

Nice-to-Have Deliverables:

1. **Block type filtering**, which is useful, but this is just an enhancement to the basic component search so it's not completely necessary for MVP.
2. **Block usage reference panel** because a dedicated panel showing all usages would be convenient, but it's not essential if the basic search works well enough.
3. **Bookmarking functionality** is helpful for experienced users, but it's not critical for the core search functionality, therefore not MVP.
4. **Search history and saved searches** would be a quality of life improvement that could be added later, but also not MVP.

Timeline and Milestones

Phase 1: Research and Planning (3 weeks)

- Study App Inventor's block structure and workspace implementation
- Review existing search implementations in similar platforms
- Design search algorithms and data structures
- Create UI mockups and user flow diagrams

Phase 2: Core Implementation (8 weeks)

- Implement block indexing system
- Develop basic search functionality
- Create UI components for search interface
- Implement result highlighting and navigation

Phase 3: Testing and Refinement (3 weeks)

- Performance testing with large projects
- User testing and feedback collection
- Bug fixes and performance optimization
- Implementation of additional features if time allows

Phase 4: Documentation and Finalization (2 weeks)

- Create user documentation
- Prepare technical documentation for future maintenance
- Final code review and cleanup
- Submit completed project

Related Work

I researched several existing approaches to inform my implementation:

Scratch offers basic search functionality but lacks component-specific filtering. I'll build on their approach but with deeper integration specific to App Inventor's needs.

Traditional IDEs like VS Code and IntelliJ have sophisticated search systems. I'll adapt their indexing and navigation concepts to work in a block-based environment.

App Inventor's current navigation between blocks and procedures provides a foundation I can extend with search capabilities.

My solution differs by directly addressing the unique challenges of block-based programming, particularly finding component usage within collapsed procedures - a specific pain point mentioned by users.

Biographical Information

I currently work as a Lead Teacher of Computer Programming, teaching HTML, CSS, JavaScript, and Python to middle school students. Before this, I was a Software Engineer at American Family Insurance for 3.5 years. In that role, I:

- Developed web applications using React, Redux, TypeScript, and Node.js
- Refactored legacy codebases to improve maintainability
- Built front-end components and integrated with backend services
- Implemented analytics systems similar to the indexing needed for this project

I've also worked as a Teaching Assistant for web development courses at both Correlation One and Stack Education. This experience helped me understand how to explain complex programming concepts clearly.

My combination of technical skills in JavaScript and Java plus my teaching experience gives me a unique perspective for creating a search system that is technically sound and intuitive for users at different skill levels.

Time Commitment

I can commit 30 hours per week to this project. My teaching position allows flexible scheduling outside class hours. I have no planned vacations or other commitments that would interfere with this schedule.

I'm located in the Eastern Time Zone (UTC-4) and available for regular meetings with mentors as needed.

Communication

I will maintain regular communication through your preferred channels:

- Updates on GitHub issues and pull requests
- Weekly progress reports via email or chat
- Participation in scheduled video calls
- Prompt responses to feedback (within 24 hours)

I understand clear communication is essential for open source development and commit to being transparent about my progress and any challenges I encounter.