Tutorial mode for Swift in the VS Code Extension

Contact details

Full name:Deep Bartaria
Email:deepbartaria4141@gmail.com
Community ID:DecoderDeep
Location (Country and city):Mathura, Uttar Pradesh, India
Webpage / Blog / Microblog / Portfolio : https://github.com/deepbartaria
https://www.linkedin.com/in/deepbartaria

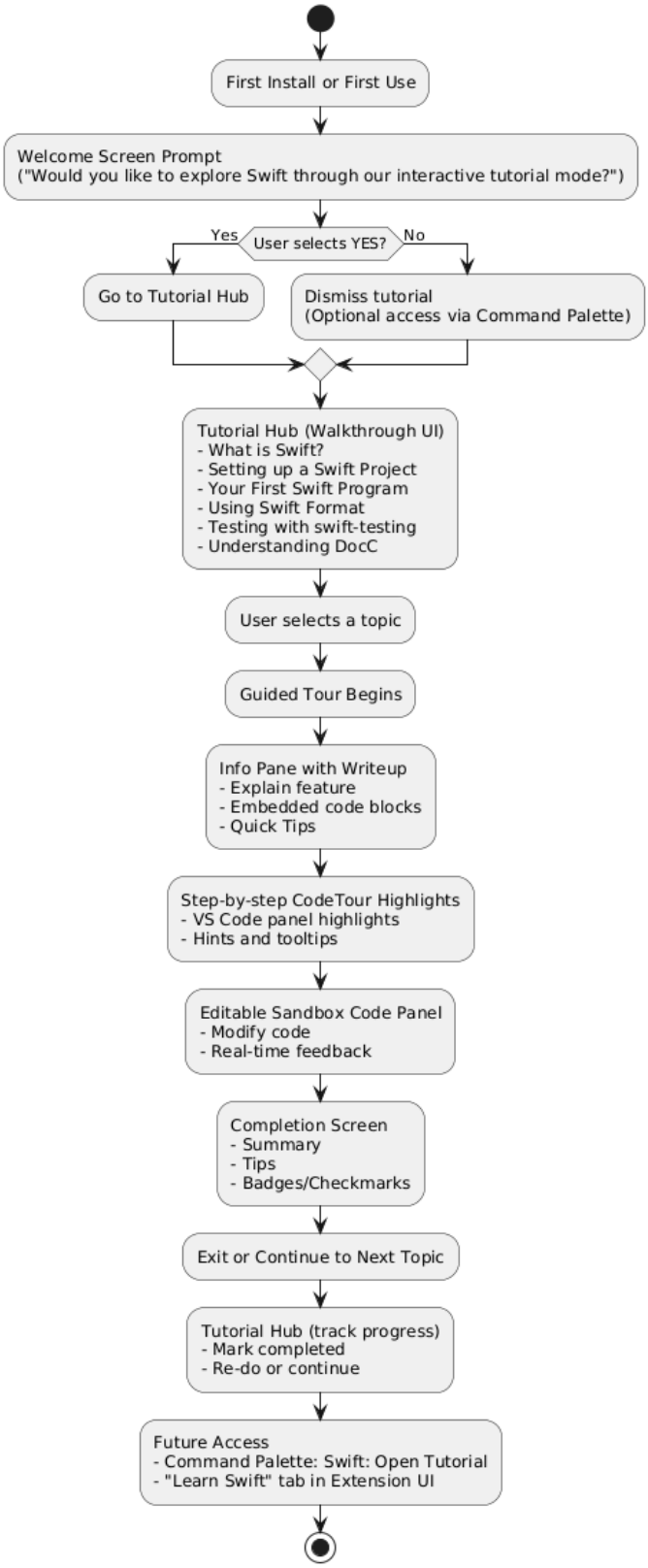
Project synopsis

The proposed project, “Tutorial mode for Swift in the VS Code Extension,” aims to enhance the onboarding experience for developers—especially beginners—who are using the Swift programming language within Visual Studio Code. Swift is widely known for its safety, performance, and developer-friendly syntax. However, many new users who install the VS Code Swift extension may be unfamiliar with the language or programming altogether. This project seeks to introduce a comprehensive, interactive tutorial mode integrated within the VS Code environment to guide users through core Swift concepts, tools, and features.

This tutorial mode will utilize features from the VS Code Walkthrough API or integrate with the CodeTour extension. It will include curated, hands-on examples adapted from resources such as the Swift Book, DocC tutorials, swift-testing documentation, and swift-format guidelines. By building this mode into the VS Code extension, users will be able to experiment and learn in a guided, project-based environment without ever leaving the IDE.

The expected outcome is a better learning curve for Swift, improved adoption and retention of users, and a smoother introduction to the Swift ecosystem—especially in educational or beginner-friendly environments.

Flow Diagram



Project in detail

This project focuses on the design and development of an interactive tutorial module embedded directly within the VS Code Swift extension. The tutorial will provide users with real-time coding tasks, explanations, and feedback as they explore Swift features and tooling.

The project will be executed in multiple stages:

1. **Research and Planning**: Understand the existing Swift extension codebase, capabilities of Walkthrough and CodeTour APIs, and survey beginner pain points in learning Swift.

2. **Curriculum Design**: Identify a sequence of beginner-friendly modules derived from Swift documentation (e.g., variables, control flow, functions, optionals, testing, formatting).

3. **Content Development**: Write markdown/code files that guide the user, integrating them into walkthroughs or CodeTour steps.

4. **Extension Integration**: Modify the existing VS Code Swift extension to optionally show the tutorial mode on first install or through a command.

5. **Testing and Iteration**: Gather feedback from sample users, iterate on usability, flow, and content difficulty.

6. **Documentation and Deployment**: Provide contribution guides for adding new tutorial modules, and include documentation for users.

Project implementation and timeline

Minimal set of deliverables

- A working tutorial mode within the Swift VS Code extension.

- At least 3-5 beginner-level interactive tutorials covering basics, testing, and formatting.
- Integration with either the VS Code Walkthrough API or the CodeTour extension.
- A system to allow contribution of new tutorials in the future (i.e., template or content schema).

Additional ‘if time allows’ deliverables (optional)

- User progress tracking via extension state.
- Unlockable achievements or badges.
- Integration with online Swift documentation.
- Theme customization for accessibility.

Detailed timeline

Week	Task
May 20 - May 26	Understand VS Code Swift extension architecture, Walkthrough/CodeTour APIs
May 27 - June 2	Research Swift documentation sources, identify topics for tutorials
June 3 - June 9	Create content structure (lesson format, markdown/code layout) coding
June 10 - June 16	Implement first prototype of Walkthrough or CodeTour integration
June 17 - June 23	Develop tutorials for variables, types, functions, control flow
June 24 - June 30	Develop tutorials for swift-testing and swift-format
July 1 - July 7	Feedback loop: usability testing with peers, refine tutorial UX
July 8 - July 12	Mid-term evaluation

July 13 - July 20	Implement improvements based on mentor feedback
July 21 - July 27	Add advanced tutorials and optional progress-tracking logic
July 28 - August 3	Polish integration and handle edge cases in different OS/dev setups
August 4 - August 10	Buffer period for fixing bugs, optimizing UI and UX
August 11 - August 17	Final documentation, contributor guide, testing
August 18 - August 25	Final testing, prepare for release and wrap-up blog

---

Communication with mentors

I will maintain regular communication with my mentors through weekly Google Meet check-ins, asynchronous discussion via Slack/Discord or email, and blog updates. Every week, I will report:

- What I accomplished during the week.
- Plans for the upcoming week.
- Any challenges or blockers I encountered.

These updates will ensure transparency and continuous feedback throughout the project.

Past Experiences

I previously participated in **GirlScript Summer of Code (GSSoC)**, where I contributed to open-source projects and collaborated with mentors and other contributors. This experience gave me a solid understanding of the open-source development workflow, including issue tracking, pull requests, and code reviews.

I also have strong experience with **TypeScript**, which I use regularly in my web development projects. I've built several full-stack applications using modern web technologies like **React**, **Node.js**, and **Express**, and have hands-on experience with integrating APIs, managing state,

and building interactive UIs. My familiarity with TypeScript will be especially valuable for working on VS Code extension development, where typed JavaScript is crucial for maintaining robust and scalable code.

These experiences have prepared me well to contribute meaningfully to this GSoC project and adapt quickly to the requirements of a complex codebase.

Motivation

As a student who has often been overwhelmed by new languages or toolchains, I deeply understand the need for beginner-friendly guidance. Building a tutorial mode within VS Code for Swift can truly lower the barrier for entry. Not only does this align with my passion for education technology, but it also challenges me to think about pedagogy, UX, and extension architecture simultaneously.

Additionally, this project helps me expand my knowledge of Swift, TypeScript, and VS Code internals—technologies I am highly interested in, and i think i have every thing that is essential for this project like a Macbook. I also see this as a way to contribute meaningfully to the Swift open-source ecosystem and empower future developers.