AUTO CODE WALKER

Stop reading code. Start experiencing it.

TABLE OF CONTENTS

THE ONBOARDING BOTTLENECK

We’ve all been there!
joining a new project, staring at thousands of lines of code, feeling completely lost.

Traditional onboarding is broken!

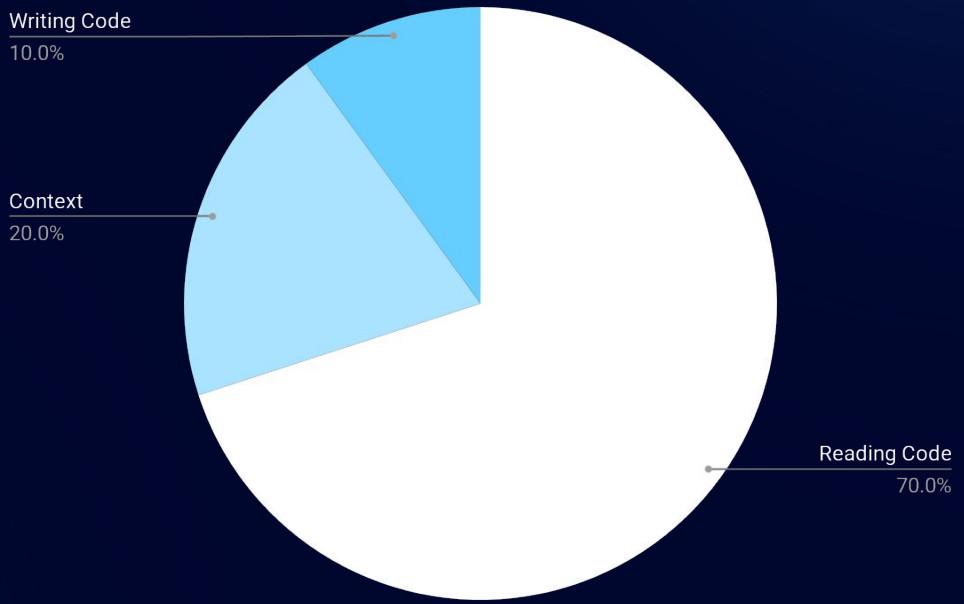
It forces you to choose between struggling alone or interrupting your team.

WHY IT STAYS BROKEN?

- **Context Trap** - New hires spend weeks getting into the code
- **Senior Tax** - Senior lose hours explaining same logic over and over
- **Doc Rot** - External Documentations is disconnected from code and always outdated

WHERE THE TIME GOES?

- Reading Code
- Context Switching
- Writing Code



IN DEPTH ANALYSIS
AUTO CODE WALKER

Where time gets lost?

Jumping between files, class and function definitions... What developers need first?: entry points, key methods/classes, data moving through the system...

“Where do I start” problem

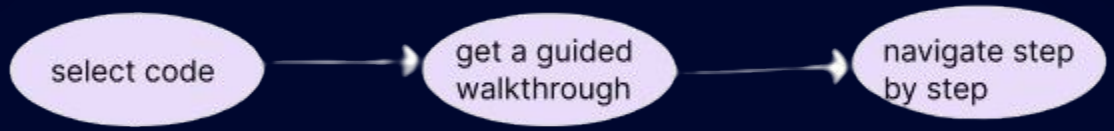
In complex projects, the biggest time sink isn’t writing code - its **understanding existing code**

BEHIND THE SCENES

We started with a simple question: How to help developers and make something useful for them?

We curate what to extract (PSI structure, key identifiers) and shape it into explanations that remain grounded in real code

From plugin architecture and PSI extraction, to walkthrough generation and UI presentation, we aligned on one goal: **Reduce confusion and speed up understanding**



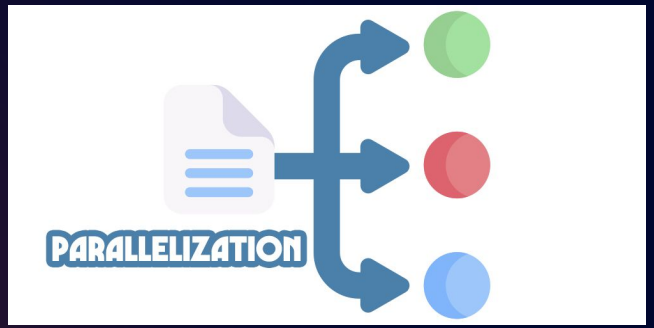
TECH STACK & ARCHITECTURE

PSI TREE

We used IntelliJ PSI to extract code structure and map each walkthrough step to exact source locations.

JAVA PARALLELIZATION

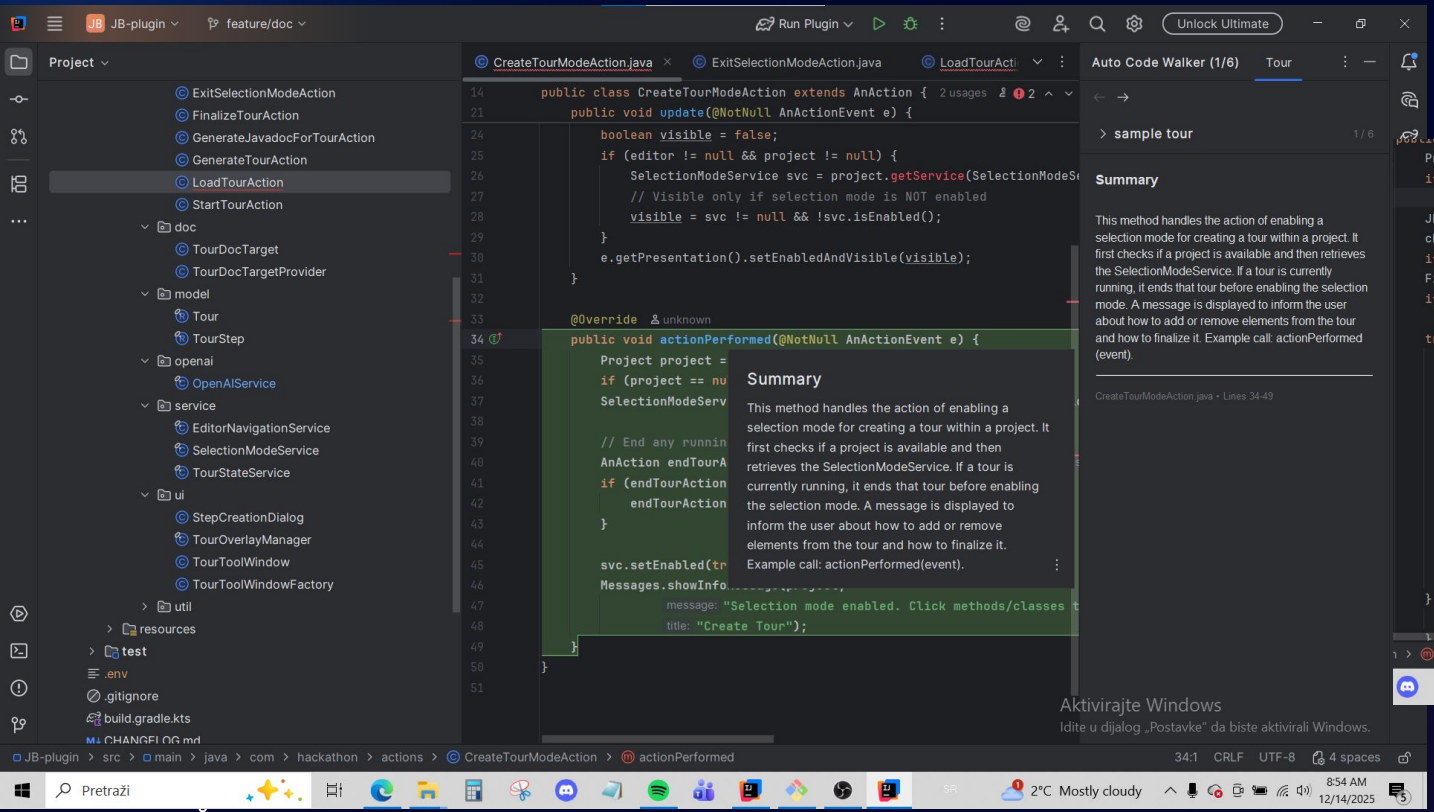
We used Java background threads to run OpenAI requests without freezing the IDE



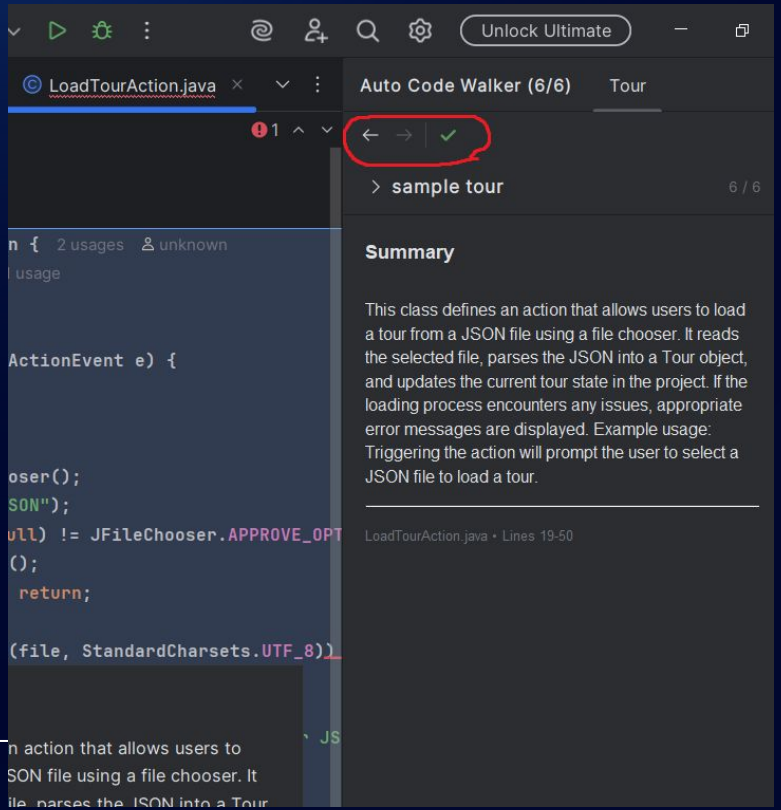
OPEN AI API

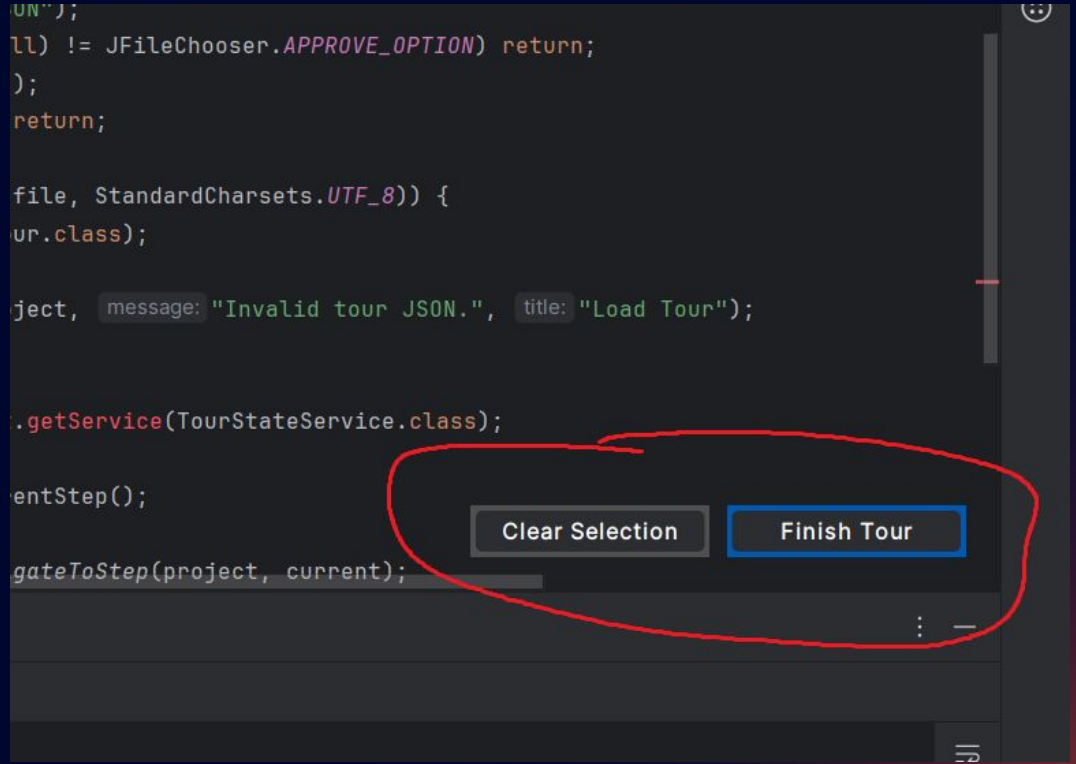
We integrated OpenAI by sending the selected PSI extracted code, so the model returns summarization

DEMO SCREENSHOTS

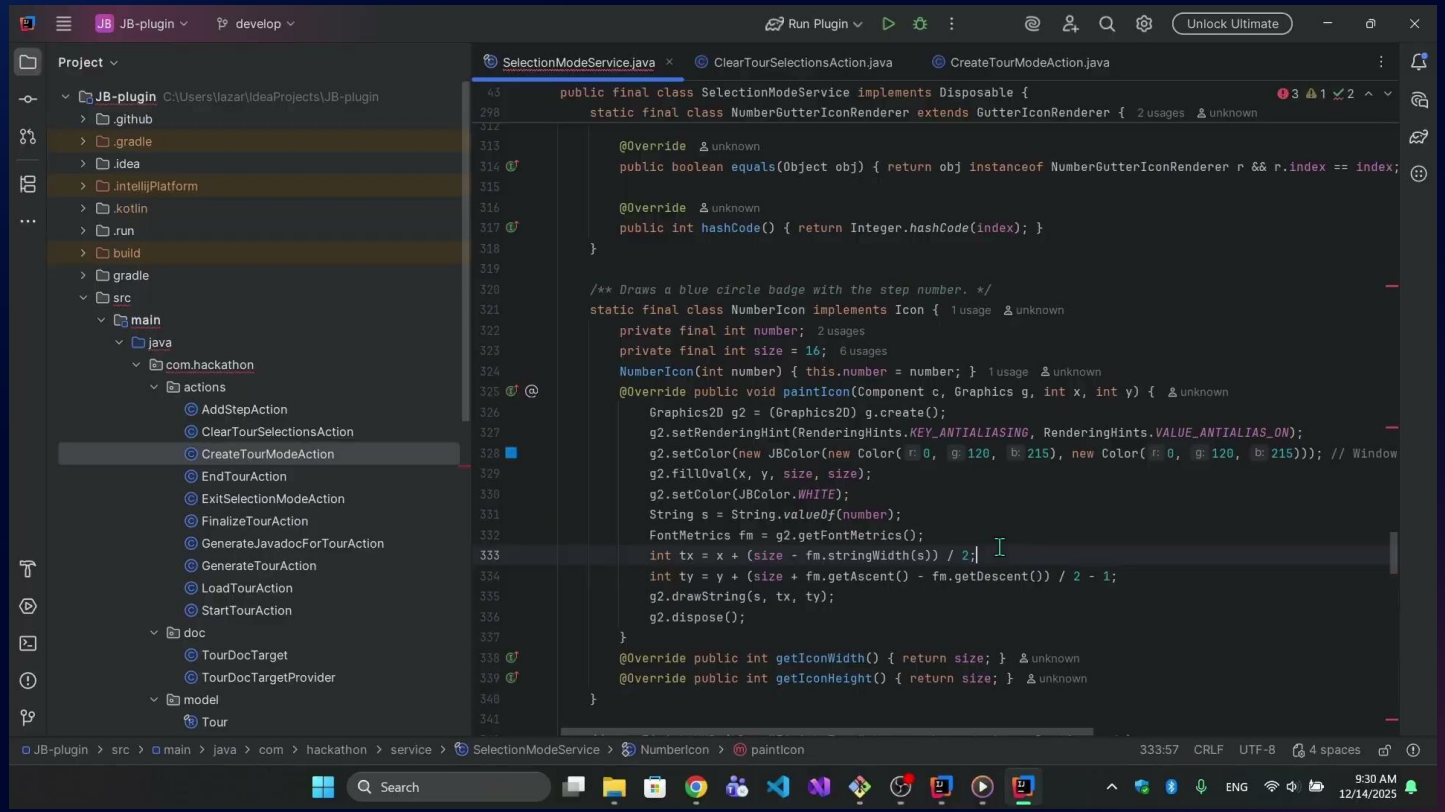


DEMO SCREENSHOTS





DEMO VIDEO


JB-plugin
develop
Run Plugin
Unlock Ultimate
Project
JB-plugin C:\Users\lazar\IdeaProjects\JB-plugin
.github
.gradle
.idea
.intellijPlatform
.kotlin
.run
build
gradle
src
main
java
com.hackathon
actions
AddStepAction
ClearTourSelectionsAction
CreateTourModeAction
EndTourAction
ExitSelectionModeAction
FinalizeTourAction
GenerateJavadocForTourAction
GenerateTourAction
LoadTourAction
StartTourAction
doc
TourDocTarget
TourDocTargetProvider
model
Tour
SelectionModeService.java
ClearTourSelectionsAction.java
CreateTourModeAction.java
public final class SelectionModeService implements Disposable {
static final class NumberGutterIconRenderer extends GutterIconRenderer { 2 usages unknown
@Override unknown
public boolean equals(Object obj) { return obj instanceof NumberGutterIconRenderer r && r.index == index;
@Override unknown
public int hashCode() { return Integer.hashCode(index); }
}
/** Draws a blue circle badge with the step number. */
static final class NumberIcon implements Icon { 1 usage unknown
private final int number; 2 usages
private final int size = 16; 6 usages
NumberIcon(int number) { this.number = number; } 1 usage unknown
@Override public void paintIcon(Component c, Graphics g, int x, int y) { unknown
Graphics2D g2 = (Graphics2D) g.create();
g2.setRenderingHint(RenderingHints.KEY_ANTIALIASING, RenderingHints.VALUE_ANTIALIAS_ON);
g2.setColor(new JBColor(new Color(r: 0, g: 120, b: 215), new Color(r: 0, g: 120, b: 215))); // Window
g2.fillOval(x, y, size, size);
g2.setColor(JBColor.WHITE);
String s = String.valueOf(number);
FontMetrics fm = g2.getFontMetrics();
int tx = x + (size - fm.stringWidth(s)) / 2;
int ty = y + (size + fm.getAscent() - fm.getDescent()) / 2 - 1;
g2.drawString(s, tx, ty);
g2.dispose();
}
@Override public int getIconWidth() { return size; } unknown
@Override public int getIconHeight() { return size; } unknown
}
JB-plugin > src > main > java > com > hackathon > service > SelectionModeService > NumberIcon > paintIcon
333:57 CRLF UTF-8 4 spaces
Search
ENG
9:30 AM
12/14/2025