

# IDEaL

An IDE for all Learners

# Hello World!



**Jin-Hee**

Designer



**Monique**

PM



**Neel**

Engineer



**Amrita**

PM/Engineer



## Passion for CS Education

~10 years total experience as tutors, SLs, TAs, and head TAs, with a focus on intro CS courses



## Diversity in Experience

Prior academic and internship experience in product management, design, and SWE roles



## Software Development Background

Experience developing large software projects in a variety of languages, as well as tackling CS security + AI model development challenges

# Our Team

# Problem

**Learning to code can be a frustrating, overwhelming, and discouraging experience.**

Pain Points:

- Incomprehensible error messages
- Complicated, feature-heavy IDEs
- Domain-specific jargon → high barrier to entry

Desires:

- Prioritization of code style/efficiency
- Help in learning language idioms



“I was nervous going into [intro] classes because I feel like I don’t have a ton of CS background, and everyone else does.”

“It can be tough to figure out why my computer is struggling to run programs - is it something inefficient in my code?”

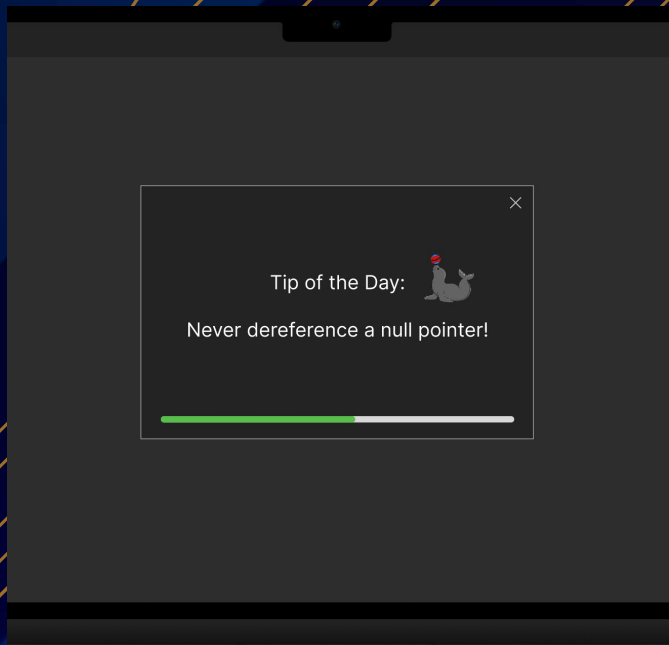
“I don’t want an error message that just tells me what line an error is at...Name the variable. Say specifically what is bad.”

# Solution

**IDEaL - An educational IDE for all programming learners.**

We make the process of learning to code less overwhelming, providing students with an understandable, encouraging, and fun way to program. Our solution offers:

- Support with deciphering opaque errors when debugging
- Simple and easy-to-use interface
- Learn-as-you-go integrated programming lessons/resources



# Happy Path

When coding... I can see what isn't working... Have the understanding to fix it... Then fix it and learn!

```
karel.cpp 1 /*
merge.cpp 2 * karel.cpp
sierpinski.cpp 3 * This file contains a recursive function that finds the number of
4 * steps it will take our character Karel to get home.
5 */
6
7
8
9
10 // * Takes the initial x, y of Karel
11 * Returns the minimum number of
12 * in order for Karel to get home
13 */
14 int karel(int x, int y) {
15     if (x == 0 || y == -1) {
16         return 0;
17     }
18     return karel(x, y);
19 }
20
```

```
karel.cpp 1 /*
merge.cpp 2 * karel.cpp
sierpinski.cpp 3 * This file contains a recursive function that finds th
4 * steps it will take our character Karel to get home.
5 */
6
7
8 // * Takes the initial x, y of Karel's starting position.
9 * Returns the minimum number of steps it takes
10 * in order for Karel to get home to 1, 1.
11 */
12 int karel(int x, int y) {
13     if (x == 1 && y == 1) {
14         x++;
15     } else {
16         karel(x - 1, y - 1);
17     }
18 }
19
20
```

**Build Errors**  
Line 15: You can't include numbers immediately after ++. See [C++ Operators](#) for more details.

### C++ Operators

<b>Increment</b> var++ ++var	Returns the value of var, then increases that value by 1. Increases the value of var by 1, then returns that value.
<b>Decrement</b> var-- --var	Returns the value of var, then decreases that value by 1. Decreases the value of var by 1, then returns that value.

Logical Operators >

```
11 // * in order for Karel to get home to 1, 1.
12 */
13 int karel(int x, int y) {
14     if (x == 1 && y == 1) {
15         x++;
16     } else {
17         karel(x - 1, y - 1);
18     }
19 }
20
```

**Build Errors**  
Line 15: You can't include numbers immediately after ++. Rememb

var++ increments after return  
var-- increments before return

```
karel.cpp 1 /*
merge.cpp 2 * karel.cpp
sierpinski.cpp 3 * This file contains a recursive function that finds the number of
4 * steps it will take our character Karel to get home.
5 */
6
7
8 // * Takes the initial x, y of Karel's starting position.
9 * Returns the minimum number of steps it takes
10 * in order for Karel to get home to 1, 1.
11 */
12 int karel(int x, int y) {
13     if (x == 1 && y == 1) {
14         x++;
15     } else {
16         return karel(x - 1, y - 1);
17     }
18 }
19
20
```

# Key Product Features

- Simple, working IDE with the following learning-friendly features
- Error messages with "plain English," beginner-friendly wording
- On-the-fly style tips so you can improve style as you're coding
- A place to review conceptual material that's relevant to the course/current assignment
- A way to keep "sticky notes" for key learnings

# Market Fit (TAM/SAM/SOM)

- There were **26.8 million** active software developers in the world at the end of 2021.
- Obtainable market: students studying CS at Stanford.
  - ~10% of students who take CS106A/B → ~**360 people**.
- ~1200 students per quarter across both classes → ~3600 for the whole year.



# Competitive Landscape

- Main competitors in the space include:
  - VSCode
  - Sublime Text
  - JetBrains
- Data mining across VSCode has been subject to criticism (CoPilot)
- No one focuses on beginning programming (yet)

# Business Model - Prosumer



## Community

Free offering with key features (e.g. friendly errors, style feedback, crowdsourced lessons). Students contribute to feedback dataset

**\$0**



## Institutional

Adds limited course integration with grading, student submission storage, course resources, integrity checkers (e.g. MOSS)

**\$2.50 per student per course annually**



## Institutional+

Adds full course integration with live TA debugging session support, analytics on course and assignment performance

**\$5 per student per course annually**

# Projected Revenue



## Community

Free!



## Institutional

Assuming 20 courses at each university and ~200 students/course

\$10,000/yr/school



## Institutional+

Assuming 20 courses at each university and ~200 students/course

\$20,000/yr/school

# Traction



## Right now...

- A handful (~40-50) CS106 students we've interviewed have expressed strong interest in this support from an IDE
- No formal plan / signup
- Raised discussion with Stanford's intro CS faculty → interest
  - Growing frustration with IDEs like Qt already

## Looking ahead...

- Maintain discussion with Stanford CS faculty
- Beta testing on CS106 students
  - Test and make improvements before official partnerships
- Introduce to section leaders \*
- Branch out to intro CS at other institutions nearby, likely Cal, SCU

# Projections

- Current: proof of concept shown, interviews done, revenue models built out
- In next 6 months: build out MVP, ship as VSCode / Sublime extension
- In next 12 months: secure funding, build standalone application for macOS, Linux, and Windows
- In next 18 months: Sell to consumers, universities. Ship V2 of product with enhanced feature set and specific settings for university clients

# Funding and Resource Request

## Needs:

- Engineering staff: 3 engineers to start
- Marketing to universities: 5 employees on marketing team
- Design of application: 3 designers working on MVP to pitch to investors

Estimate in seed funding: \$3M

# Appendix



# Ethical Concerns

## **Quality of education**

- What is the source of the educational content? Is it true, up-to-date, etc?

## **Partnering with universities / learning institutions**

- Who ultimately controls the content on IDEaL? What if we disagree?

## **Accessibility**

- When implementing the technological features such as sticky notes or supplemental error messages, can they be found and read by a text-to-speech, for example?

## **Traditional pedagogy and progress**

- Are we assuming that our way of learning will work for everyone instead of working the other way around?



# Role Prototype (Cards)

## Learning Card

Strategyzer

Insights Name: **style/conceptual errors**

Date of Learning: **11/6**

Person Responsible: **Monique Ouk**

STEP 1: HYPOTHESIS

We believed that

style and conceptual errors matter to new CS learners as much as functionality

STEP 2: OBSERVATION

We observed

The interviewee focused on functionality first and foremost, simply because that's what they know how to do correctly. They would go back to their code after it ran to check their style.

STEP 3: LEARNINGS AND INSIGHTS

From that we learned that

Conceptual interviewers (C) "You can do this in a few ways, remember, it's for help should look like this." "I would be super helpful, especially because the interviewee prefers fixing things as they go instead of having to go back and look at everything again to fix their style."

STEP 4: DECISIONS AND ACTIONS

Therefore, we will

Include conceptual errors that appear in this role as the interviewee can't do code and have a greater understanding of what they're doing (or expected to be doing) as they write it.

Copyright Strategyzer AG

The makers of Business Model Generation and Strategyzer

## Learning Card

Strategyzer

Social Name

11/5 of Learning

Person Responsible

STEP 1: HYPOTHESIS

We believed that

the social aspect of learning how to code is NOT a top priority in a learning-based IDE.

STEP 2: OBSERVATION

We observed

strong dependence on human resources in order to help in the coding process, desire to seek help from others, awe of "really smart" students and interest in interacting with them.

STEP 3: LEARNINGS AND INSIGHTS

From that we learned that

The social aspect of learning how to code, especially in the first few weeks, is a high priority

STEP 4: DECISIONS AND ACTIONS

Therefore, we will

Implement a "Social" feature as part of our POC / prototype(s)

Copyright Strategyzer AG

The makers of Business Model Generation and Strategyzer

## Learning Card

Strategyzer

Insights Name

setting up IDE

Date of Learning: **11/7**

Person Responsible

STEP 1: HYPOTHESIS

We believed that

people know how to set up an IDE / tools

STEP 2: OBSERVATION

We observed

everything spoken with a lot of conviction; very familiar and confident in the steps that he says he takes

STEP 3: LEARNINGS AND INSIGHTS

From that we learned that

He uses the internet the most to help set up the IDE. He gets stuck on anything. Even when there are multiple guiding resources (documentation, a video, a official lesson), he would only resort to the internet because if he couldn't find the solution on google.

STEP 4: DECISIONS AND ACTIONS

Therefore, we will

make a set-up process that has explanations if needed, but is adaptable for those who don't want to read

Copyright Strategyzer AG

The makers of Business Model Generation and Strategyzer

## Learning Card

Strategyzer

Insights Name

Lectures on the go

11/7 of Learning

Person Responsible

STEP 1: HYPOTHESIS

We believed that

students would like to learn programming with lectures as they go while coding, rather than separately from the IDE experience.

STEP 2: OBSERVATION

We observed

Frustration around finding lecture materials

STEP 3: LEARNINGS AND INSIGHTS

From that we learned that

People are interested in fast, ideally "instant access" to the conceptual materials that would be most helpful to them at the time of the assignment

STEP 4: DECISIONS AND ACTIONS

Therefore, we will

Try to include lecture links / resources in a well-organized way from our IDE

Copyright Strategyzer AG

The makers of Business Model Generation and Strategyzer

## TEST CARD 1

Social aspect of learning

We believe that...

the social aspect of learning how to code is not a top priority in a learning-based IDE.

Critical: 2/3

To verify that, we will...

Create a mock version of "social features": articles notes, progress check, social IDE editing, progress check, and a "social" feature. Offer up a simple, however, on the screen, then ask students to interact in a way that would help them learn the function. Designated word "social" and are encouraged to ask questions.

Test Card: 1/3

Created at 11/11/2023 in IDEs (Programming)

Date Reliability: 7/5

Where in the social features based on...

1) number of clicks

2) number of questions, in what the feature does for them

we are right if...

students do not click in the social features window

students do not ask about the social aspects of learning

## TEST CARD 3

style and conceptual errors

We believe that...

style and conceptual errors matter to learners as much as functionality

Critical: 3/3

To verify that, we will...

Create a sample of social code snippets in a social IDE. Create a sample of social code snippets in a social IDE. Create a sample of social code snippets in a social IDE. Create a sample of social code snippets in a social IDE. Create a sample of social code snippets in a social IDE.

And measure...

average, ranking of style and conceptual errors in completed code

we are right if...

students do not automatically read style and conceptual errors as less important / frequent / lower priority than functional errors.

## TEST CARD 4

IDE only

We believe that...

people know how to set up an IDE / tools

Critical: 2/3

To verify that, we will...

present 3 scenarios in a paper of a social IDE. Create a sample of social code snippets in a social IDE. Create a sample of social code snippets in a social IDE. Create a sample of social code snippets in a social IDE. Create a sample of social code snippets in a social IDE.

And measure...

average, ranking of style and conceptual errors in completed code

we are right if...

students are able to troubleshoot, and quickly set up a paper in at least one of the provided scenarios

## TEST CARD 5

"plain English" messaging

We believe that...

people learning to code appreciate simplicity "plain English" phrasing in their error messages / suggestions

Critical: 3/3

To verify that, we will...

take 3 actual error messages from the IDE. Create a sample of social code snippets in a social IDE. Create a sample of social code snippets in a social IDE. Create a sample of social code snippets in a social IDE. Create a sample of social code snippets in a social IDE.

And measure...

average, ranking of style and conceptual errors in completed code

we are right if...

students' messaging of error messages and ability to fix the error

## LEARNING CARD 1

understandable error messages

WE BELIEVED THAT...

people who are learning to code appreciate simplicity, "plain English" phrasing in their error messages / suggestions

WE OBSERVED...

even debugging experience of the new error message

FROM THAT WE LEARNED...

students' messaging of error messages and ability to fix the error

THEREFORE WE WILL...

from our message length, as a criterion of evaluation for a good message

## TEST CARD 2

learn - design - go coding

We believe that...

students would like to learn programming as they go while coding rather than separately from the IDE experience

Critical: 2/3

To verify that, we will...

have 3/100 students with 2 specific questions (similar to the first question, but with a different focus) answer material they have recently learned. In the first function, they will use a "social" IDE, in the second they will use a social IDE with design.

And measure...

average, ranking of style and conceptual errors in completed code

we are right if...

students do not click in the social features window

# Role Prototype (Insights)

## **Our top 3-5 synthesized insights:**

- Error readability can greatly affect a beginner's motivation level/dedication to continuing CS.
- On-the-fly conceptual reminders are very helpful, especially because our interviewees liked to fix their code as they go instead of having to go back at the end to fix things.
- People want fast/"instant" access to conceptual material that would be most helpful to them for their assignment.

# Role Prototype

Our product's value comes from being a text editor that lets you learn as you go. A typical user could be a CS106A student. When the 106A student starts an assignment using our IDE, they'll have immediate access to the lectures and course notes relevant to the assignment they're currently working on. As they begin coding, the IDE will highlight any style issues alongside a suggestion on how to improve it, which allows the user to fix their style in the process, instead of having to go back and potentially getting lost in their own code. Any errors will also be flagged and explained in "readable", plain English, which lets the user thoroughly understand the mistake they made, and how to avoid repeating that problem. The user has the option of making "sticky notes" to jot down anything they want to remember, such as a mistake they've formerly made, or specific syntax they constantly forget, etc. Weak signals that indicate our product's value would include some usage of the sticky notes. Although the user may not be using the sticky notes frequently, the fact that they're utilizing the sticky feature *at all* indicates that the user does want the option to keep their own notes (in their own words) in addition to the help that the IDE provides. Strong signals that indicate value include high retention in the IDE (e.g., not frequently clicking out or for long periods of time) since the user no longer has to heavily google the meaning of error messages and less repeated style issues, which means that the user is actively reading the style suggestions and correctly implementing the advice as they move forward. After all of our assumption and experience prototype testing, the features we will include in our GTM strategy include 1) readable error messages, 2) style suggestions, 3) relevant conceptual notes, 4) sticky note features. These four key features were proven to make the IDE valuable for the user, as they effectively assist the user's learning without being overwhelming.

# IDEaL - calm, supportive, educational

## Colors

Dark Mode



Light Mode



## Icons

example syntax "sticky notes"

For Loops (By Range)

```
for (int i = 0; i < 5; i++) {  
  // something with i  
}
```

For-Each Loops

```
for (char c: str) {  
  // something with c  
}
```



example error message

! You can't do A with B. Try checking C or review recursion.



## Textures

Inter / Inter

for notes, text

Fira Code

for the code

Ledger

for learning

## Fonts

The background is a dark blue gradient with diagonal stripes of a slightly lighter blue. Scattered throughout are small white dots and thin, light blue lines. In the top right and bottom right corners, there are stylized, light blue paperclip icons.

# Look and Feel Prototype Demo and Regular Figma

# Design Justification

## **Design**

Looking to popular IDEs with a slick look such as VSCode and Sublime, we stuck with a simplistic IDE layout with just the basics on the screen to start with: the file you're in, the code you're writing, the line numbers, and the buttons you need to make stuff run.

We also used popular learning companies such as Khan Academy or Duolingo to motivate our color selection in our style tile, ultimately opting for a mostly black/white/gray/blue palette with touches of green and yellow.

In an IDE, seeing errors is the most important thing, and we didn't want too much color to take away from that. Thus, most of the screen is dark gray / white depending on the user's selection of "dark mode" or "light mode."

## **Product Branding**

Since the learning-focus is what sets us apart from other IDEs, it made sense to heavily emphasize this in our branding: from the product name to what to put on the loading screen as the app was firing up to how we will brand ourselves in pursuit of customers.

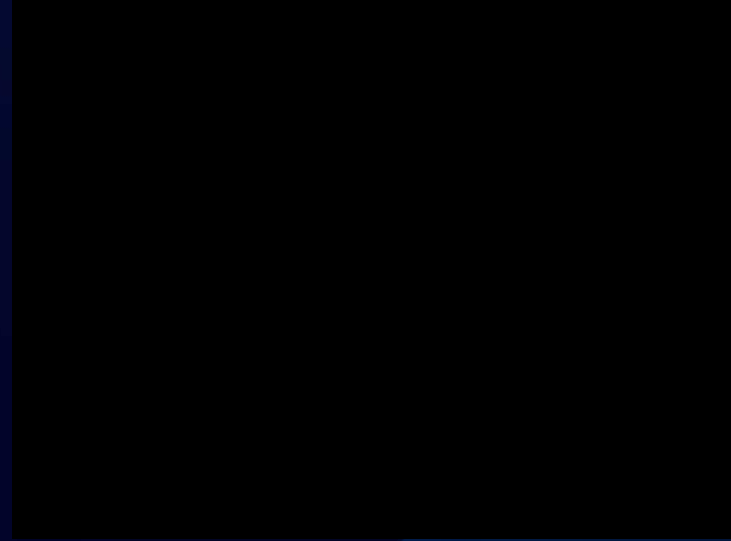


# Implementation Prototype Demo

**Tech Stack:** TypeScript, VSCode Extension API, Python, Shell scripting

## **Justification:**

Our spike prototype showcases a key feature of our product - automated error message translation. As an MVP, we created a VSCode extension with this component of our idea, leveraging the VSCode Extension API to demonstrate feasibility of our spike feature alone rather than a full standalone IDE. The extension detects any errors in student-written Python code and presents the user with a simple, understandable version of this error with debugging advice for beginner coders.



# Implementation Prototype Writeup

- Started by building a CLI tool to parse error messages into simpler, more readable messages
  - Focus on 3-4 of the most common Python/C++ error messages to start
- Take CLI tool and package into a VSCode extension
- Use HTML and CSS to add graphics to make the popup more engaging
- Test in development environment to make sure specific error messages are being parsed correctly



# Go-To Market Plan

## **MVP**

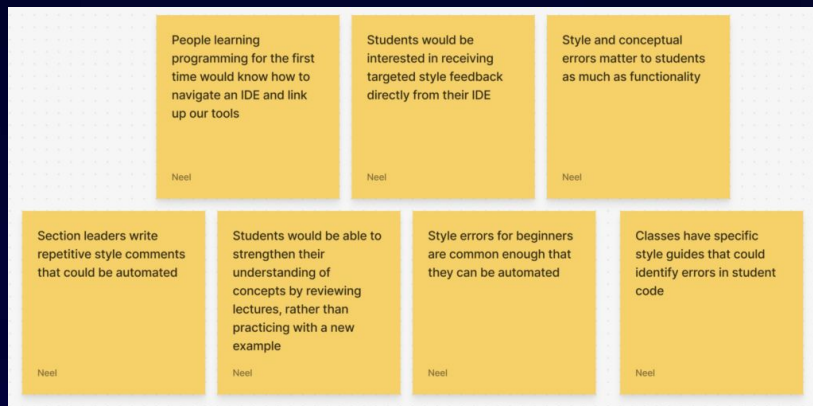
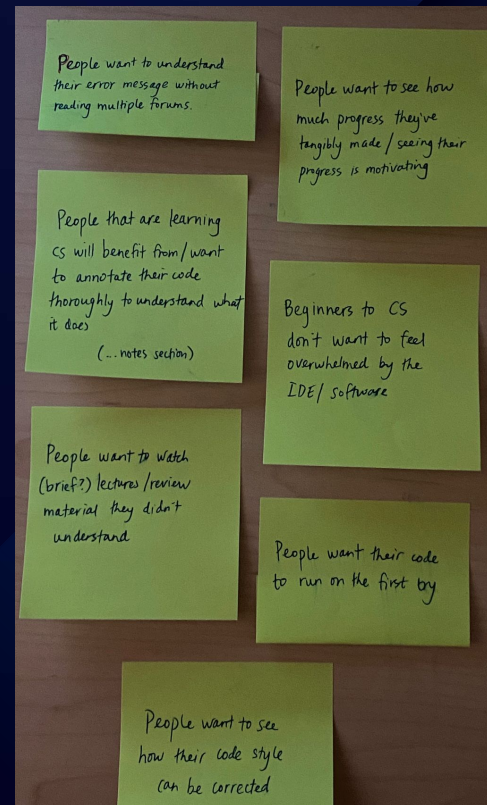
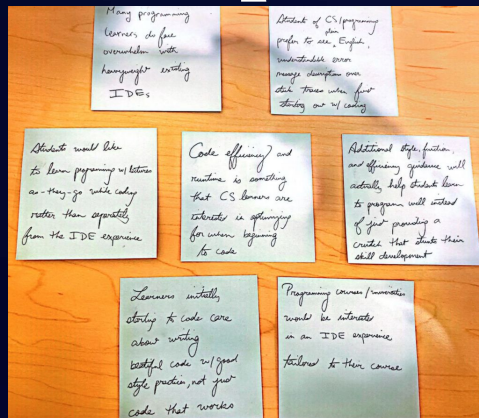
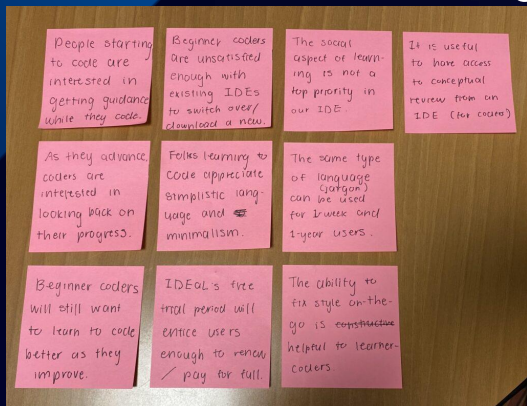
- Simple, working IDE with the following learning-friendly features:
  - Error messages with "plain English," beginner-friendly wording
  - On-the-fly style tips so you can improve style as you're coding
  - A place to review conceptual material
  - A way to keep "sticky notes" for key learnings

## **Customers, Customers, Customers!**

- We will build strong, feedback-friendly relationships with our starting partnership (Stanford CS program) to build up our reputation
  - As we expand, get connections from faculty members
- Student and faculty word-of-mouth
- Converting customers: discuss time saved for graders, office hours, students who are frustrated
- Retaining customers:
  - Quarterly check-ins with programs to see what's working / not
  - Expanding team of engineers to ensure high + improving performance

[illegible]

# Key Assumptions

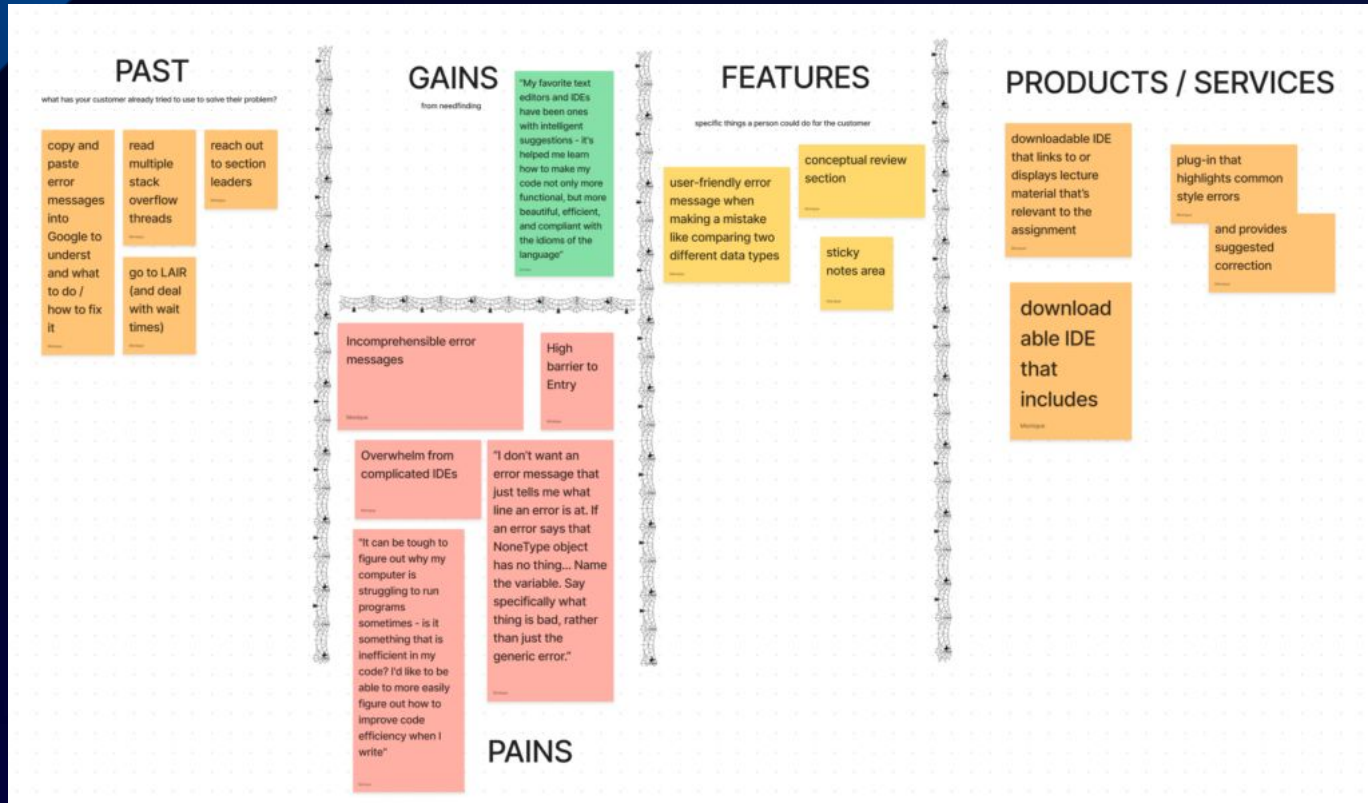




# BMC



# VPC



# All Tests/Interviews

By Team Member Conducted:

Jin-Hee + Monique:

- Priority testing with 2 individuals: Sanjay, Victoria
- Experience prototype tests with 6 individuals: Natalie, Tom, Sarah, Mason, Sam, Andrew

Jin-Hee:

- Needfinding interviews with 5 individuals: Momo, Michael, JD, Isabelle, Grant

Monique:

- Needfinding interviews with 5 individuals: Arthur, Sarah, Brian, Zoe, Claire

Neel:

- Needfinding interviews with 4 individuals: Julie, Cynthia, Lucia, Tobey
- Experience prototype testing with 2 individuals: Danny, Andrea

Amrita:

- Needfinding interviews with 4 individuals: Sumer, Megan, Anisha, Catherine



Thank you!