Potential Interview Questions:

Background/demographic questions: (10-15 minutes)

- How long have you been teaching coding?
- What motivated you to bring coding into your curriculum?
- Have you received any specific training or professional development in teaching coding?
- Do you teach programming as a stand-alone topic or integrated into other domains?
    - If so, which domains?
- Please describe the lesson structure- how often do students meet in a week? About how long? Is it a required course or an elective? During the school day or an after-school club? Grade levels?
- Which programming environment(s) do you use in your classroom?
    - Why did you choose this environment?
    - Overall, do you think this is a good application to use to introduce coding to beginners? What ages would you think this IDE is best suited for?
- Do you have experience with text-based programming IDEs?
    - Do you think it is easier for students to learn in one modality vs the other?
    - Are there any features in the block-based IDE you wish were available in text based? Vice versa?

User experience questions: (15-20 minutes)

- What is the general student sentiment about these lessons?
    - Do they like learning code, is it perceived as difficult, etc.?
    - Are students generally motivated and engaged during coding lessons?
    - Are there any particular features or aspects of the programming environment that students find especially appealing or challenging?
- Are there specific units or tasks that students seem to struggle with more than others?
    - Any common mistakes you see made?
- Overall, do you think students understand which command they need to perform a certain action?
    - Do they know where to find the commands they need and how to place them in the workspace?
    - Do they know how to combine the commands in intentional ways to reach their desired effect?
- How independent are the students in creating programs?
    - Do you tend to do more structured or unstructured projects?
    - If a student makes mistakes in their code, are they able to debug themselves or do they get stuck?
    - Are there any specific tools in the application that support students being able to correct their mistakes?
- Does the application meet the needs of the learners? Does the application meet the needs of the classroom?
    - Is there a tool or feature that would make it easier for students to learn programming concepts?
    - Is there a tool or feature that would better facilitate teaching these concepts?

Usability heuristics questions: (15-20 minutes)

- Visibility of system status:
    - How well does the IDE provide feedback and indicate progress to students as they work on their programs?
    - Do students have a clear understanding of the current state of their program while using the IDE?
- Match between system and the real world:
    - Are the terminologies and concepts used in the IDE consistent with what students have learned?
    - Do the icons, buttons, and symbols within the IDE represent familiar programming concepts or actions?
- User control and freedom:
    - How much flexibility do students have in experimenting and exploring within the IDE?
    - Are the features and commands available for students to accomplish their desired tasks?
- Consistency and standards:
    - Are the layout and organization of the IDE consistent across different sections or views?
    - Are students able to continue work outside of class (on other types of devices, operating systems, etc.)?
- Error prevention and Recognition rather than recall (already covered)
- Flexibility and efficiency of use:
    - Are there multiple ways to accomplish a task, such as keyboard shortcuts, multiple menus, touchscreen, etc.? Do the students use these shortcuts often?
    - Can students customize the IDE based on their preferences, such as changing themes or layout configurations?
- Aesthetic and minimalist design
    - Does the IDE prioritize important information and actions while minimizing unnecessary distractions? (Other aspects already covered)
    - Do you find the layout visually appealing?
- Help users recognize, diagnose, and recover from errors
    - Are there any features that allow students to easily undo or correct their actions?
- Help and documentation:
    - Are there tutorials, examples, or hints available to assist students in using the application? Do you see students use the help features?
    - Is there an online community, forum, or Q&A for the programming language and/or environment? Do students use these resources?

*Follow-up questions about usability heuristics might be asked as needed and time allows, such as:

- Do you think this application supports **[**insert heuristic here**]**?
- Why or why not? Are there any specific examples that you can think of?
- Do you think **[**insert heuristic here**]** is an important quality for an IDE for kids? In what ways do you think an application could better support it?