Create a work breakdown structure:

The polished version of the task breakdown structure:

1. Collect Data on Existing Solutions and Plugins

- Analyze existing multi-line code completion solutions and plugins.

- Document their features, strengths, and weaknesses.

- Gather information on local data structures used for analysis, parsing, and context understanding.

2. Decide on Data Structure and Model Integration

- Select appropriate data structures for storing local constructs.

- Determine where the database of available local constructs will be stored.

- Decide on the storage location for the model's source code.

- Design how the model will be integrated into the IDE.

- Plan how the model will be distributed to clients for testing.

- Choose a federated learning script and model type.

- and so on.

3. Design and Create a Prototype

- Create a detailed design for the prototype.

- Develop the prototype to test the feasibility of the chosen solutions.

- Conduct preliminary tests to gather feedback and make improvements.

4. Prepare Code Snippets and Large Codebases

-Collect and prepare code snippets and large codebases for testing.

- Ensure the datasets are representative of the target use cases.

5. Test the Model Using Prepared Code Snippets

- Test the model using prepared code snippets.

- Identify and fix any issues during testing.

6. Evaluate the Model’s Performance

- Measure accuracy, recall, and precision.

- Compare the model’s performance against benchmarks.

7. Implement Federated Learning Script

- Distribute the model to clients for training.

- Collect feedback and model updates from clients.

8. Aggregate and Evaluate Models

- Pick up all the models and decide on the best one or aggregate the best features.

- Evaluate the aggregated model and apply fine-tuning if necessary.

9. Iterate and Improve the Model

- Repeat step 8 multiple times to improve the model.

- Monitor model performance and make ongoing improvements.

10. Integrate the Model into the IDE

- Ensure seamless integration of the model into the IDE.

- Create detailed user documentation and guides.

- Prepare for the public release of the plugin.

The unpolished version:

1. Collect data on existing solutions and plugins for Multi-Line Code Completion, if they exist, various data structures for collecting local data used in analysis and parsing of the code, triggers for the code completion, and context understanding by which it can make suggestions.
2. Decide on a data structure, where the database of available local constructs is going to be stored, where the source code for the model is going to be stored, how the model will be integrated in the IDE, how it will be distributed to clients for testing, deciding on a federal learning script and model type, privacy security measures and so on.
3. Design and create a prototype for testing the feasibility of the solutions decided on beforehand.
4. Prepare code snippets and large codebases for prototype testing, training and fine-tuning.
5. Test the model using the prepared code snippets.
6. Evaluate the model’s performance on the test by measuring accuracy, recall, precision.
7. Implement the federated learning script by distributing the model to clients, to train the model and provide feedback.
8. Pick up all the models, decide on the best one, or aggregate the best features of every model into one. Evaluate the model and apply some fine-tuning if necessary to redistribute the new model to the clients.
9. Repeat step 8. multiple times to improve the model.
10. Once the model is ready for public use, integrate it into the IDE.

Estimate task effort: Note that this is not an exact measure, in fact I could be way off, I would have to discuss it with the team to achieve a more accurate estimation of time.

1. Collect Data on Existing Solutions and Plugins: 3 weeks

2. Decide on Data Structure and Model Integration: 4 weeks

3. Design and Create a Prototype: 5 weeks

4. Prepare Code Snippets and Large Codebases: 3 weeks

5. Test the Model Using Prepared Code Snippets: 4 weeks

6. Evaluate the Model’s Performance: 2 weeks

7. Implement Federated Learning Script: 4 weeks

8. Aggregate and Evaluate Models: 5 weeks

9. Iterate and Improve the Model: 7 weeks

10. Integrate the Model into the IDE: 4 weeks

Identify dependencies:

Steps 1,2,3,5,6,7,8,9,10 themselves should be completed in ascending order. As for step 4, it should be completed anywhere before step 5.

Prioritize tasks: If the need to get the first results as early as possible, it makes almost everything before that a High Priority task, also the structure of the steps and dependencies makes the order stay the same, aside from step 4 which I talked about before.

1. Collect Data on Existing Solutions and Plugins: High Priority

2. Decide on Data Structure and Model Integration: High Priority

3. Design and Create a Prototype: High Priority

4. Prepare Code Snippets and Large Codebases: Medium Priority

5. Test the Model Using Prepared Code Snippets: High Priority

6. Evaluate the Model’s Performance: High Priority

7. Implement Federated Learning Script: High Priority

8. Aggregate and Evaluate Models: High Priority

9. Iterate and Improve the Model: Medium Priority

10. Integrate the Model into the IDE: Low Priority