

1 APPENDIX

The organization of the supplementary material is as follows:

- Appendix A: Split-Point Detection Algorithm;
- Appendix B: Graph Building;
- Appendix C: Supported diagram element categories for recognition.
- Appendix D: The results of the user experiment questionnaire;
- Appendix E: More examples from the user study.
- Appendix F: The Use of Large Language Models (LLMs).

A SPLIT-POINT DETECTION ALGORITHM

The split-point detection method is shown in Algorithm 1.

Algorithm 1 Split-Point Detection

Input: Skeleton \mathcal{S} , joint map \mathcal{J}

Output: Optimised split points \mathcal{P}^*

```
1:  $\mathcal{P} \leftarrow \{p \in \mathcal{S} \setminus \mathcal{J} : \theta_p < \theta_{\text{thr}}(\ell_p)\}$ 
2: for each  $s \in \mathcal{P}$  do
3:    $\lambda_s \leftarrow \exp(-\beta\kappa_s)$ 
4: end for
5:  $\mathcal{E} \leftarrow \emptyset$  // initialise empty edge set
6: for each unordered pair  $(s, t) \subset \mathcal{P}$  with  $\|s - t\|_2 < r$  do
7:    $\phi_{s,t} \leftarrow \exp(-\|s - t\|_2/\sigma)$  // compute pairwise cost
8:    $\mathcal{E} \leftarrow \mathcal{E} \cup \{(s, t)\}$  // add edge to graph
9: end for
10: Build graph  $G = (\mathcal{P}, \mathcal{E})$ 
11:  $\mathbf{e}^* \leftarrow \text{GraphCut}(\{\lambda_s\}, \{\phi_{s,t}\})$ 
12:  $\mathcal{P}^* \leftarrow \{s \mid e_s^* = 1\}$ 
13: return  $\mathcal{P}^*$ 
```

B GRAPH BUILDING

Each diagram is defined as a graph, denoted as G . Each diagram G_i ($i = 1, \dots, K$) consists of a sequence of strokes, denoted as $\{S_a^i \mid a = 1, \dots, n_i\}$, where n_i represents the total number of strokes in diagram G_i . Each stroke S_a^i possesses the following attributes: a stroke class label C_a^i , which indicates the category of the stroke, and a symbol identifier I_a^i , which represents the symbol to which the stroke belongs. Therefore, each stroke S_a^i can be represented as a graph node $N_a^i = (S_a^i, C_a^i, I_a^i)$. For the diagram G_i , the collection of all its strokes constitutes the node set $\{N_a^i \mid a = 1, \dots, n_i\}$. In addition, the connections between strokes in diagram G_i form an edge set, denoted as $\{E_b^i \mid b = 1, \dots, m_i\}$, where m_i represents the total number of edges in diagram G_i . Each edge E_b^i represents the connection between two strokes and has an edge class label C_b^i , which distinguishes between positive edges and negative edges. A positive edge indicates that the two connected nodes belong to the same instance, while a negative edge indicates that the two connected nodes belong to different instances. Each edge E_b^i can be represented as $E_b^i = (N_x^i, N_y^i, C_b^i)$, where N_x^i and N_y^i denote the two endpoints of the edge. In summary, a diagram $G_i = (V^i, E^i)$ is defined by its node set $V^i = \{N_a^i \mid a = 1, \dots, n_i\}$ and edge set $E^i = \{E_b^i \mid b = 1, \dots, m_i\}$. By accurately recognizing the node categories and edge categories, stroke-level instance recognition of offline diagrams is achieved.

C SUPPORTED DIAGRAM ELEMENT CATEGORIES FOR RECOGNITION

As shown in Table 1, our method is capable of recognizing the 20 categories of diagram elements.

D THE RESULTS OF THE USER EXPERIMENT QUESTIONNAIRE

To further validate the practical utility of our method, we conducted a questionnaire survey among the participants. We conducted a p-value test on the questionnaire results to validate the effectiveness and statistical significance of the proposed method in question-answering and editing tasks. 90% of participants agreed that the method significantly improved the accuracy of diagram-based question-answering tasks ($p < 0.001$), making responses more precise and semantically relevant. In editing tasks, 90% of participants believed the method’s significant enhancement of offline diagram reusability and its substantial improvement in editing accuracy ($p < 0.001$).




















Text	Terminal	Rounded Rectangle	Process	Preparation
<i>Best Wishes</i> <i>DiagramMaster</i>				
Predefined Process	Multiple Document	Manual Operation	Manual Input	Extract
				
External Data	Ellipse	Document	Display	Decision
				
Database	Data	Circle	Arrow	Annotation
				

Table 1: Supported symbol types.

E MORE EXAMPLES FROM THE USER STUDY

The following are the four typical types of errors that LLMs may encounter in user experiments, as shown in Fig. 1,2,3 and 4. The comparison between DiagramDiff and the results generated by LLMs is also presented.

F THE USE OF LARGE LANGUAGE MODELS (LLMs)

We conducted user experiments using LLMs to validate the effectiveness of our method, and did not use LLMs for writing articles or other purposes

108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161

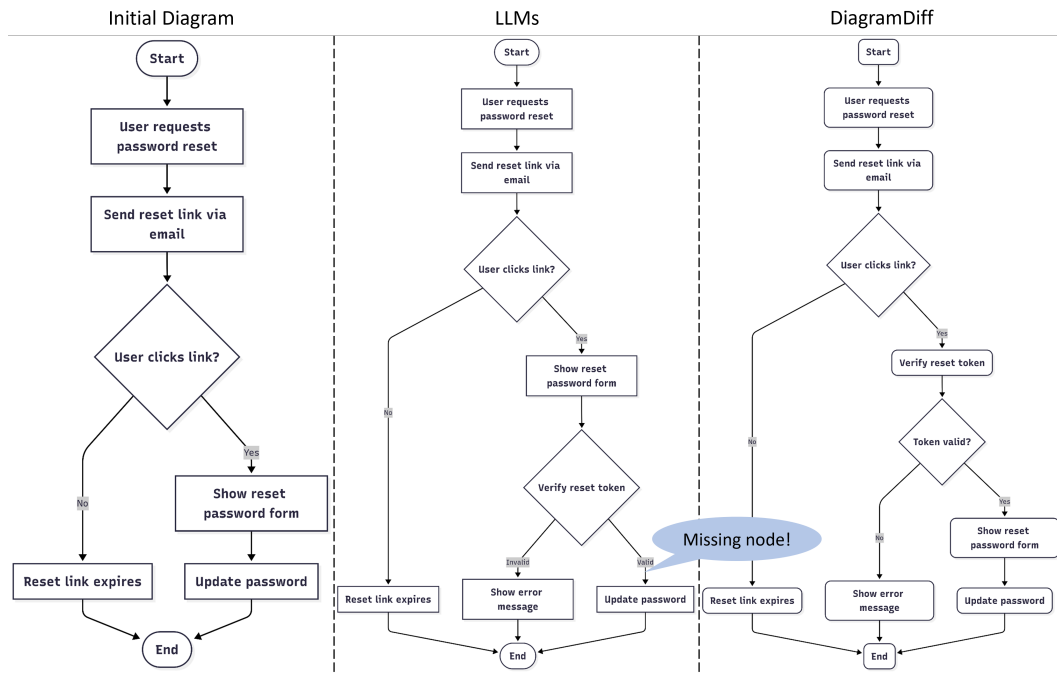


Figure 1: Order: Add a step to verify the shipping address before processing payment. If the address is invalid, notify the customer and end the process.

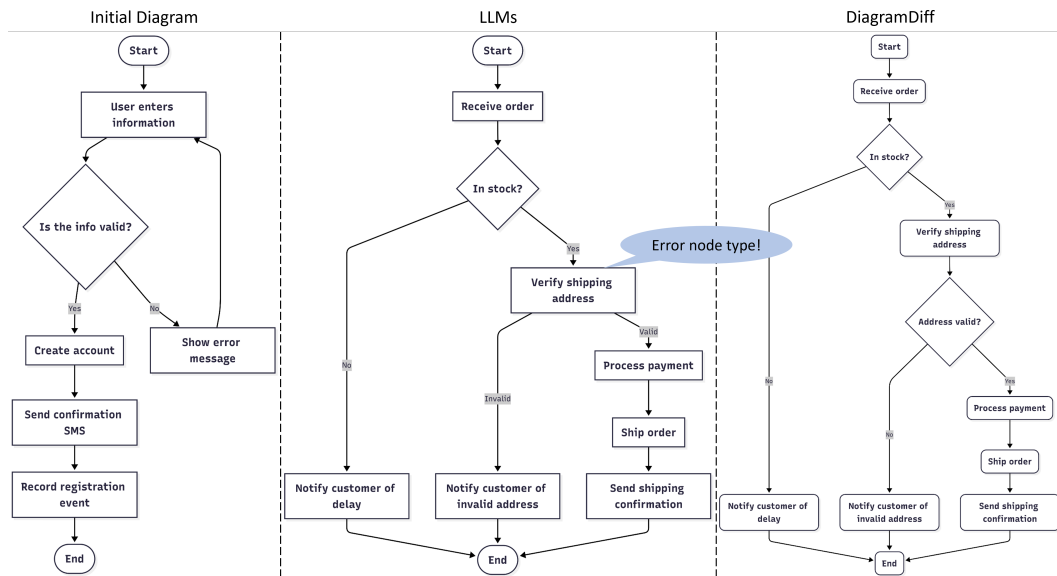


Figure 2: Order: Add a step to check if the meeting room is available after checking organizer availability. If the room is not available, propose alternative times.

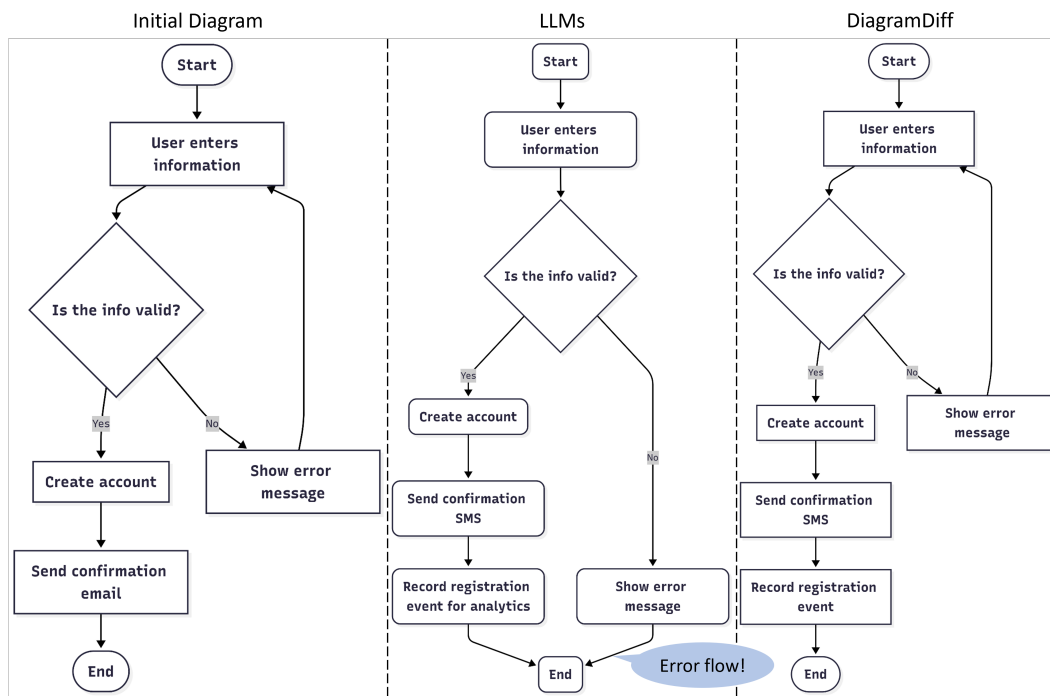


Figure 3: Order: Replace the email confirmation step with SMS confirmation, as SMS is more immediate. After sending the SMS, add a step to record the registration event for analytics.

216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269

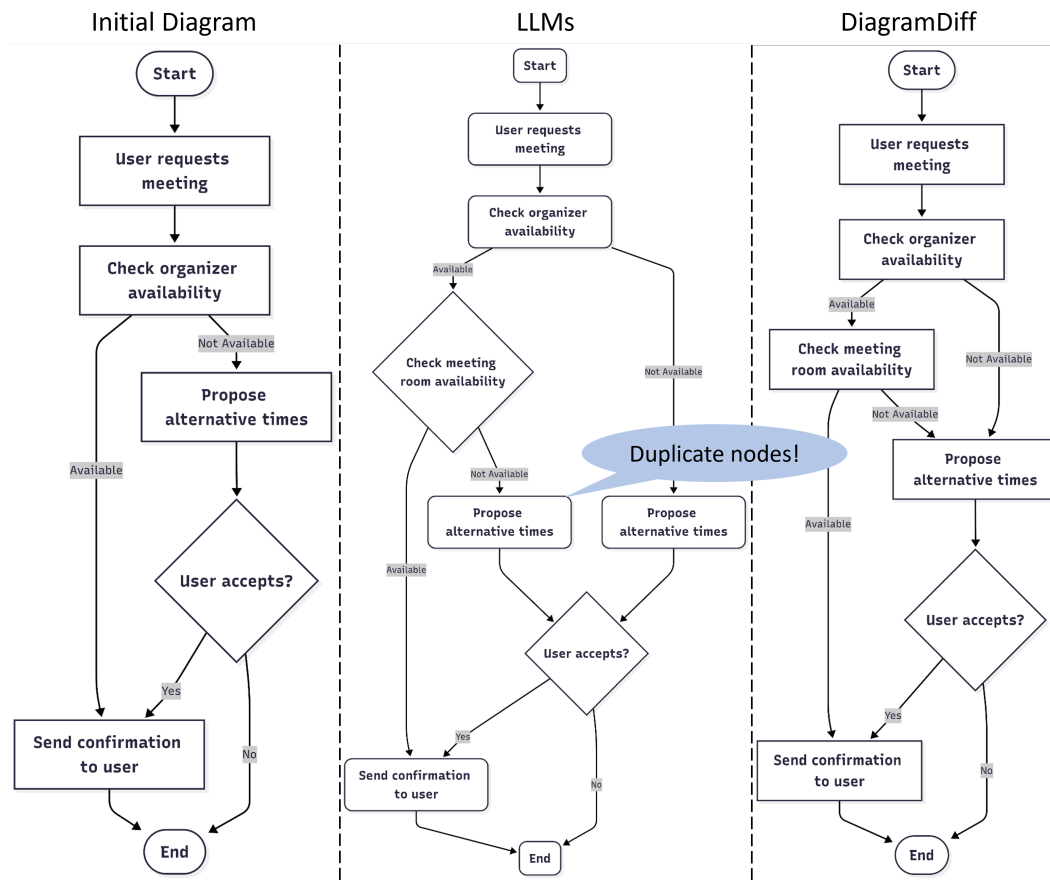


Figure 4: Order: Before updating the password, add a step to verify the reset token. If the token is invalid, show an error and end the process.