Algolia Key Features – Testing Plan

We will test the following **Algolia features one by one** under the **Search & Relevance** section.

Search & Relevance Features

1. Typo Tolerance

Algolia built-in typo tolerance Options:

- "true" (default)
- "false" (disable)
- "min" (only one typo allowed) "strict" (strict typo handling)

```
let { hits } = await index.search("iphnoe", {
  typoTolerance: "true", // "false" | "min" | "strict"
});
```

2. Synonyms

✅ **Frontend Layer – Custom Synonym Map**

```
const synonymMap = {
bug: "mug",   coffe:
"coffee",   laptp:
"laptop",   tshrt: "tshirt",
  moblie: "mobile",
};
```

✅ **Backend Layer – Synonym API**

```
app.post("/add-synonyms", async (req, res) => {   try {
    const { synonyms } = req.body;

    const synonymObjects = synonyms || [
      {
        objectID: "bug-mug",
        type: "synonym",
synonyms: ["bug", "mug"],
      },
      {
```

```
      objectID: "coffe-coffee",
type: "synonym",
      synonyms: ["coffe", "coffee"],
    },
  ];

  await index.saveSynonyms(synonymObjects, { replaceExistingSynonyms: true });
res.json({ success: true });
 } catch (err) {
  res.status(500).json({ error: err.message });
 }
});
```

3. Query Rules

More powerful than synonyms (can rewrite queries, boost results, filter, etc).

```
app.post("/add-query-rule", async (req, res) => {   try {
  const { rule } = req.body;

  const ruleObject = rule || {
objectID: "coffee-bug-to-mug",
    condition: {        pattern:
"coffee bug",
      anchoring: "is",
    },
    consequence: {
params: {
       query: "coffee mug", // query rewrite
      },
    },
  };

  await index.saveRule(ruleObject);    res.json({
success: true });
 } catch (err) {
  res.status(500).json({ error: err.message });
 }
});
```

4. Optional Words / Stop Words

Use when search has no results.

```
let { hits } = await index.search(q, {    removeWordsIfNoResults:
"allOptional",
  // options: "none" | "lastWords" | "firstWords" | "allOptional"
});
```

5. Query Suggestions ✅

Backend Layer

```
app.post("/add-suggestions", async (req, res) => {    try {
    const { suggestions } = req.body;

    const suggestionObjects = suggestions || [
{ objectID: "s1", query: "coffee mug" },       {
objectID: "s2", query: "polo shirts" },
      { objectID: "s3", query: "iphone 15" },
    ];

    await suggestionsIndex.saveObjects(suggestionObjects);
    res.json({ success: true });
  } catch (err) {
    res.status(500).json({ error: err.message });
  }
});
```

✅ **Frontend Layer**

```
let { hits } = await suggestionsIndex.search("coffee"); console.log("Suggestions:", hits);
```

Advanced Enhancements (Next Steps)

- **Vector / Neural Search** – semantic, meaning-based search.
- **Rules** – complex query rewrites, filters, boosts.
- **AI Synonyms** – automatic synonym detection via AI.
- **Collections** – group results by categories or sources.
- **Re-Ranking** – reorder results using ML models.
- **Query Categorization** – detect intent & classify query.
  Personalization – user-based search ranking.