Appendix

Appendix A - Key Code Extracts

A1 - Hybrid recommendation logic (recommendation.js)

```
// main recommendation function
function getRecommendations(itemId, activeAllergens, items, cooccurrence) {
  const selectedItem = items.find((i) => i.id === itemId);
  // return empty array if no item was found in the items.json
  if (!selectedItem) return [];

  const candidates = new Map();

  // get co-occurrences for item
  const coMap = cooccurrence[itemId] || {};
  for (const [otherId, score] of Object.entries(coMap)) {
    const item = items.find((i) => i.id === otherId);
    if (!item) {
      continue;
    }

    // if item is not allergen safe skip
    if (!isSafe(item, activeAllergens)) {
      continue;
    }

    candidates.set(otherId, {
      item,
      score,
      reason: `Often bought with ${selectedItem.name}`,
      source: "cooccurrence",
    });
  }
```

```
  }

  // if candidates are less than 3, use tag fallback
  if (candidates.size < 3) {
    for (const item of items) {
      if (item.id === itemId) {
        continue;
      }
      // if not allergen safe skip
      if (!isSafe(item, activeAllergens)) {
        continue;
      }

      // if item id is present in candidates skip
      if (candidates.has(item.id)) {
        continue;
      }

      // get similarity score by comparing tags
      const sim = tagSimilarity(selectedItem.tags || [], item.tags || []);

      if (sim > 0) {
        const tagList =
          selectedItem.tags && selectedItem.tags.length > 0
            ? selectedItem.tags.join(", ")
            : "selected";
        candidates.set(item.id, {
          item,
          score: sim,
          reason: `Matches your ${tagList} choice`,
          source: "tags",
        });
      }
    }
  }
```

Purpose: Demonstrates algorithmic complexity and ranking strategy.

A2 - Allergen filtering (recommendation.js)

```
// helper function for allergen filter
function isSafe(item, activeAllergens) {
  if (!activeAllergens || !Array.isArray(activeAllergens)) return true;
  if (!item.allergens || !Array.isArray(item.allergens)) return true;
  return !item.allergens.some((a) => activeAllergens.includes(a));
}
```

Purpose: Confirms safety filtering is enforced in the algorithm, not only in the UI.

A3 - Co-occurrence builder (cooccurrenceBuilder.js)

```
function buildCooccurrence() {
  const dataDir = path.resolve(__dirname, '../data');
  const orders = JSON.parse(fs.readFileSync(path.join(dataDir, 'orders.json'), 'utf8'));

  const counts = {};      // pair counts: itemA -> itemB -> count
  const totals = {};      // totals: itemA -> total orders containing it

  for (const order of orders) {
    const unique = [...new Set(order)];

    for (const a of unique) {
      totals[a] = (totals[a] || 0) + 1;
      if (!counts[a]) counts[a] = {};
      for (const b of unique) {
        if (b === a) continue;
        counts[a][b] = (counts[a][b] || 0) + 1;
      }
    }
  }

  const cooccurrence = {};
  for (const a of Object.keys(counts)) {
    cooccurrence[a] = {};
    for (const b of Object.keys(counts[a])) {
      cooccurrence[a][b] = counts[a][b] / totals[a]; // relative frequency
    }
  }

  fs.writeFileSync(path.join(dataDir, 'cooccurrence.json'), JSON.stringify(cooccurrence, null, 2));
  console.log('Co-occurrence recomputed.');
}
```

Purpose: Shows how relative frequency scores are computed.

A4 - Admin authentication + protected routes (auth.js)

```
function loginHandler(req, res, users) {
  // get username and password from request body
  const { username, password } = req.body;

  // check if user exists
  const user = users.find((u) => u.username === username);
  if (!user) return res.status(401).json({ error: "Invalid credentials" });

  // compare password with stored hashed password
  const ok = bcrypt.compareSync(password, user.passwordHash);
  if (!ok) return res.status(401).json({ error: "Invalid credentials" });

  req.session.user = { username: user.username };
  res.json({ success: true });
}

function requireAuth(req, res, next) {
  if (!req.session || !req.session.user) {
    return res.status(401).json({ error: "Not authorised" });
  }
  next();
}
```

Purpose: Demonstrates secure access control.

A5 - Recommendations request triggered by state changes (ItemDetail.jsx)

```
useEffect(() => {
  if (!item) return;

  setLoading(true);
  setError(null);

  fetchRecommendations(item.id, activeAllergens || [])
    .then((data) => {
      setRecs(data);
    })
    .catch((err) => {
      console.error(err);
      setError('Failed to load recommendations');
    })
    .finally(() => setLoading(false));
}, [item, activeAllergens]);

if (!item) {
  return <div className="item-detail-placeholder">Select an item to view details.</div>;
}

const toggleAllergen = (a) => {
  setActiveAllergens((prev) =>
    prev.includes(a) ? prev.filter((x) => x !== a) : [...prev, a]
  );
};
```

Purpose: Shows reactive UI updates and frontend–backend integration.

A6 - Tag update parsing + state update (AdminTagEditor.jsx)

```
const handleSave = async () => {
  if (!selectedId) return;
  const tags = tagsText
    .split(',')
    .map((t) => t.trim())
    .filter((t) => t.length > 0);

  try {
    const res = await updateItemTags(selectedId, tags);
    const updatedItem = res.item;

    // update items in parent
    const newItems = items.map((i) =>
      i.id === updatedItem.id ? updatedItem : i
    );
    onItemsUpdated(newItems);

    setMessage({ type: 'success', text: 'Tags updated successfully.' });
  } catch (err) {
    console.error(err);
    setMessage({ type: 'error', text: 'Failed to update tags.' });
  }
};
```

Purpose: Shows maintainability and live updates without code edits.

A7 - Centralised API calls (api.js)

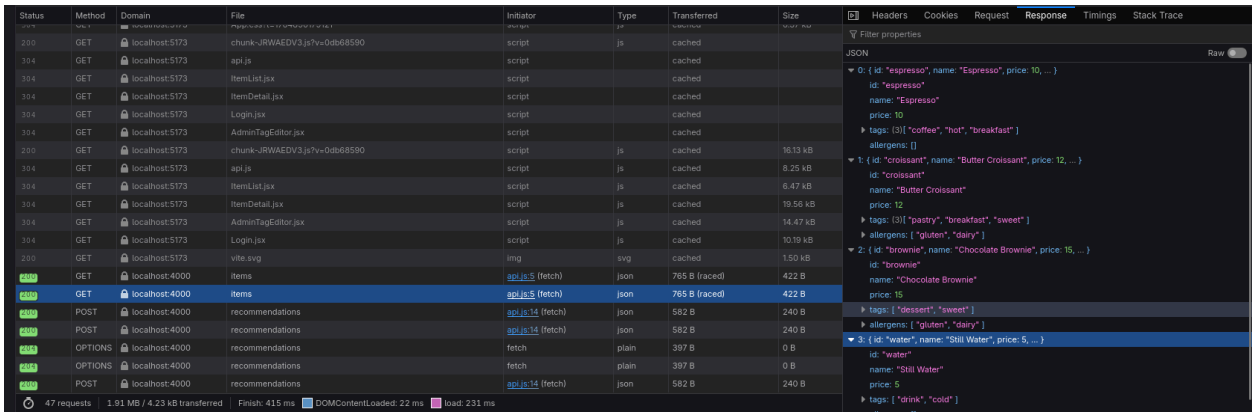
```
export async function fetchRecommendations(itemId, activeAllergens) {
  // send item id with active allergens in body and get recommendations
  const res = await fetch(`${BASE_URL}/api/recommendations`, {
    method: 'POST',
    credentials: 'include',
    headers: { 'Content-Type': 'application/json' },
    body: JSON.stringify({ itemId, activeAllergens }),
  });
  if (!res.ok) throw new Error('Failed to fetch recommendations');
  return res.json();
}
```

```
export async function updateItemTags(itemId, tags) {
  // update tags in an item by admin
  const res = await fetch(`${BASE_URL}/api/admin/items/${itemId}/tags`, {
    method: 'POST',
    credentials: 'include',
    headers: { 'Content-Type': 'application/json' },
    body: JSON.stringify({ tags }),
  });
  if (!res.ok) throw new Error('Failed to update tags');
  return res.json();
}
```

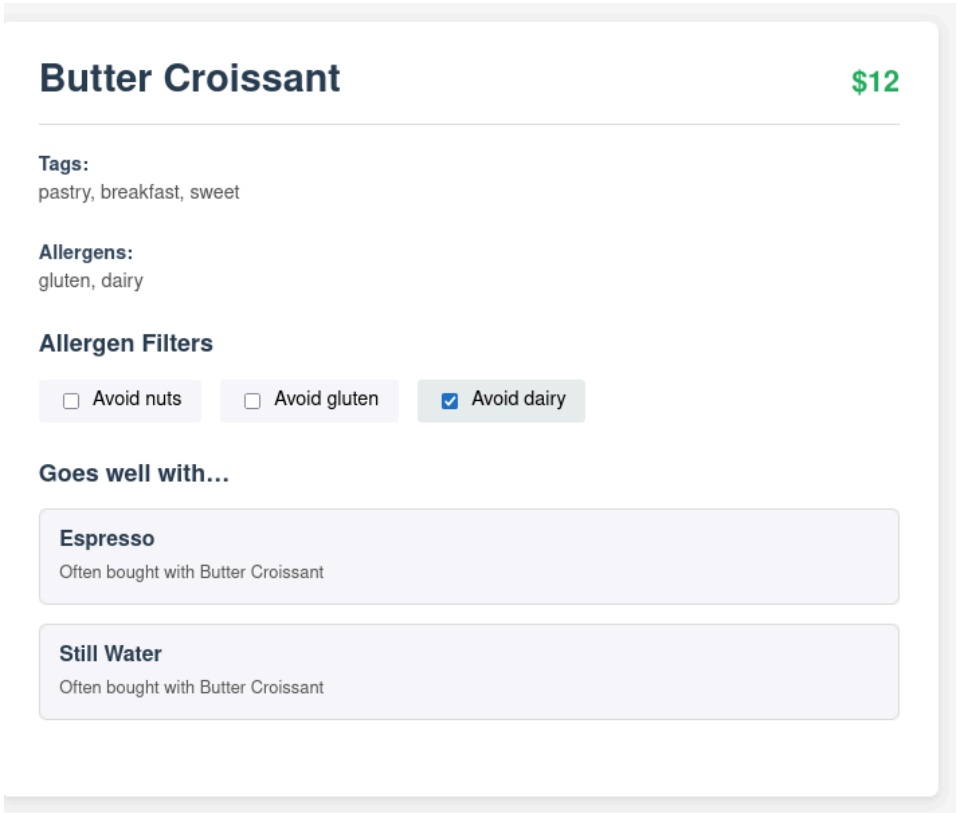
Purpose: Shows abstraction and separation of concerns.

Appendix B - Test Evidence

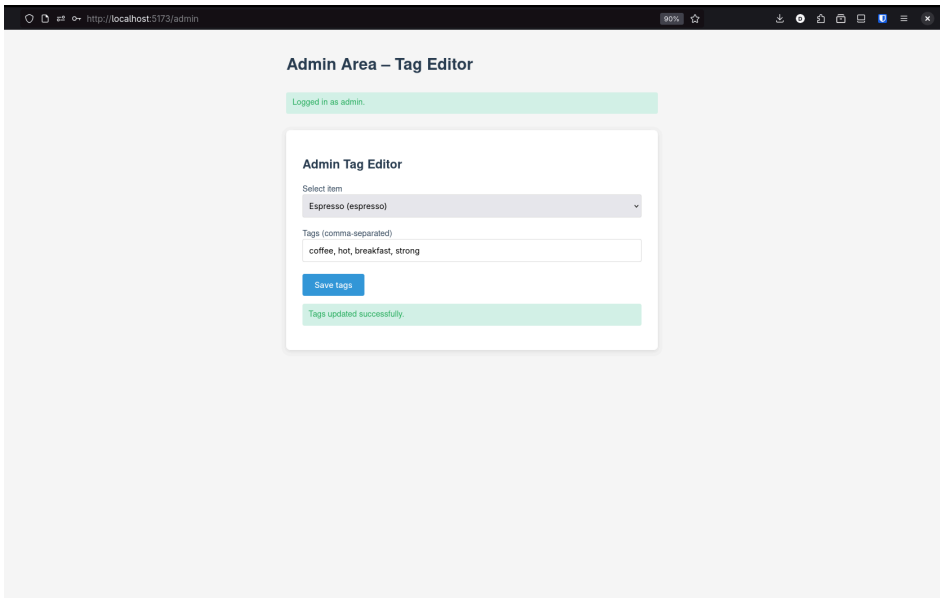
B1 - Performance testing output



B2 - Allergen filter test



B3 - Admin tag update test



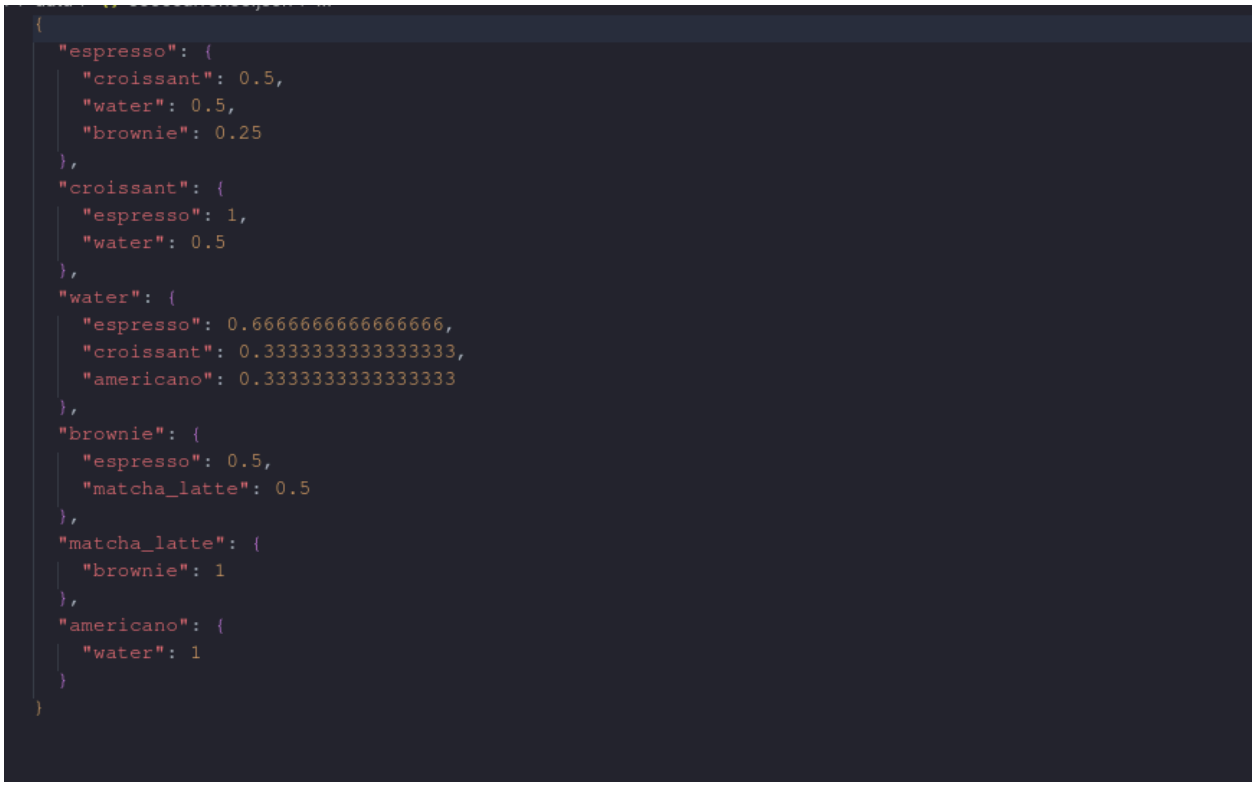
Appendix C - Data Files Used

C1 - Example item format (items.json)


```
[
  {
    "id": "espresso",
    "name": "Espresso",
    "price": 10,
    "tags": [
      "coffee",
      "hot",
      "breakfast",
      "strong"
    ],
    "allergens": []
  },
  {
    "id": "croissant",
    "name": "Butter Croissant",
    "price": 12,
    "tags": [
      "pastry",
      "breakfast",
      "sweet",
      "croissant"
    ],
    "allergens": [
      "gluten",
      "dairy"
    ]
  },
```


C2 - Example order format (orders.json)

C3 - Example co-occurrence format (cooccurrence.json)



```
{
  "espresso": {
    "croissant": 0.5,
    "water": 0.5,
    "brownie": 0.25
  },
  "croissant": {
    "espresso": 1,
    "water": 0.5
  },
  "water": {
    "espresso": 0.6666666666666666,
    "croissant": 0.3333333333333333,
    "americano": 0.3333333333333333
  },
  "brownie": {
    "espresso": 0.5,
    "matcha_latte": 0.5
  },
  "matcha_latte": {
    "brownie": 1
  },
  "americano": {
    "water": 1
  }
}
```

Appendix D - Running the Product

Environment: Node.js

To run backend:

1. cd Product/server
2. yarn install
3. yarn start

To run frontend:

1. cd Product/client
2. yarn install
3. yarn run dev

Admin access:
Username: admin
Password: *123qwe*

Appendix E - Client Consultation Evidence

E1 - Initial Email (Outreach)

From: Amj1.alhashemi@gmail.com
To: lucille.thebrothers@gmail.com

Subject: IB Computer Science Project - Menu Recommendation System
Date: 14 August 2025

Dear Lucille,

I'm Ali, an IB Computer Science student who's working on my Internal Assessment project. I am assigned with the task of developing a software solution for a real client and I believe that The Brothers LLC would be a perfect client to be provided with a digital menu system which would be a better solution to the current menu system in place. It would be great to discuss if there are any challenges or improvements you would want to see in how your menu would look to customers. Most importantly around the topic of personalized suggestions and allergen awareness.

Would you be available for a short meeting this week at any time at your convenience?
Thank you for your time.

Regards,
Ali

E2 - Client Reply

From: lucille.thebrothers@gmail.com
To: Amj1.alhashemi@gmail.com

Subject: Re: IB Computer Science Project - Menu Recommendation System
Date: 16 August 2025

Hi Ali,
Thanks for reaching out. Yes, there are a few things that we wanted to improve in our menu for a while now. The system in place is quite basic and our customers sometimes take a while to decide or even have issues where they are not aware of allergens in items they ordered.
Happy to meet. Are you free Thursday afternoon around 3pm at the cafe?

Thanks,
Lucille
Manager, The Brothers LLC

E3 - Post-Meeting Follow-Up

From: Amj1.alhashemi@gmail.com
To: lucille.thebrothers@gmail.com
Subject: Summary of Requirements - Menu Recommendation System
Date: 23 August 2025

Dear Lucille,
Thank you for taking the time to meet with me on Thursday. As discussed, I have summarised the key requirements below for your confirmation:

1. This system would recommend matching items when a customer selects a menu item.
2. The allergen information would be shown clearly and any recommendation that includes a customer's allergen will be excluded from the recommendation.
3. Staff would be able to update the item's tags and metadata through an admin web application without needing the developer to make changes.
4. The system will work through a web browser on the existing tablets without the staff devices needing to install a new mobile application.

Could you confirm whether this accurately reflects what we discussed? I will use this as the basis for my success criteria.

Thanks again,
Ali

E4 - Client Confirmation

From: lucille.thebrothers@gmail.com
To: Amj1.alhashemi@gmail.com

Subject: Re: Summary of Requirements - Menu Recommendation System
Date: 24 August 2025

Hi Ali,
Yes that looks right. Some of the points like point 2 are especially important for us since we want to make sure we are not recommending something a customer has marked as an allergen concern. Moreover, point 3 is something we really need because currently any change has to go through someone technical which is slow.

Looking forward to seeing what you build.

Best regards,
Lucille

E5 - Consultation Meeting Notes

Date: 22 August 2025
Location: The Brothers LLC, Abu Dhabi (main branch)
Attendees: Ali (student developer), Lucille (cafe manager / client)
Duration: Approximately 40 minutes

Discussion Points:

- The current system has a static digital menu with no recommendation engine. Allergen info is displayed statically but did not include any filtering
- Client pain point 1: Customers spend a long time browsing resulting in slower table turnover
- Client pain point 2: Two incidents in the past year where a customer ordered an item not realising it contained an allergen they had mentioned verbally to staff
- Client pain point 3: Any update to item tags or metadata requires calling a developer and the client staff cannot make changes themselves
- Solution should work on existing tablets via a browser (no new app installs)
- Admin access should be password-protected; only manager-level staff should edit metadata

Technical Constraints:

- Hardware: Existing Android tablets
- Software: System must run in a modern browser and client uses Wi-Fi at the café
- Data access: Client can provide sample order history and full POS integration is out of scope
- Security: Admin panel must require authentication and customer side must be read-only

Agreed Next Steps:

- Present a draft formal problem statement and success criteria
- Client to provide a sample list of menu items with prices and allergen information
- Follow-up email to confirm requirements summary

Appendix F - Client Evaluation Feedback

F1 - Evaluation Request

From: Amj1.alhashemi@gmail.com
To: lucille.thebrothers@gmail.com

Subject: Menu Recommendation System - Completed, Requesting Feedback
Date: 15 January 2025

Dear Lucille,
I hope you are doing well. I am happy to inform you that the menu recommendation system we discussed back in August is completed now. I have also recorded a screen recording of the system and a test link which I have attached to this email so you can see it for your reference and without need of setting up

anything from your end. To help with my assessment, could you let me know if the following were met from your perspective:

1. Recommendations appeared quickly enough for a café environment
2. The explanation shown with each recommendation was clear to understand
3. The allergen filter correctly excluded unsafe items from suggestions
4. The admin login was secure and worked as expected
5. You were able to update item tags without needing technical help
6. Changes to tags updated the recommendations immediately without restarting the system

If you have any comments on each point it would be very helpful.

Thank you again for your time throughout this project.

Regards,
Ali

F2 - Client Feedback

From: lucille.thebrothers@gmail.com
To: Amj1.alhashemi@gmail.com

Subject: Re: Menu Recommendation System - Completed, Requesting Feedback
Date: 16 January 2025

Hi Ali,

I watched the recording and it looks good. Here are my thoughts for each point you mentioned:

1. Yes, the suggestions came up straight away when an item was selected without any noticeable delay which is important since our customers do not like waiting on the tables.
2. The “often bought with” text is easy enough to understand.
3. This one is the most important for us. I tested the dairy filter and the items with the dairy did not show up and this is exactly what we needed especially after the incidents we mentioned when we first met.
4. The login worked fine.
5. I was able to change the tags myself after watching the recording once and I did not need to call anyone.
6. I changed a tag and selected the item again and the recommendation had already been updated. I did not expect it to be that quick

Overall it covers what we asked for at the start. The allergen side and the admin panel are the two things that matter most to us practically and both work the way we discussed.

Best Regards,
Lucille,
Manager, The Brothers LLC