Brisbane Infinity X App Specification

June 4, 2025

1 Overview

1.1 Layman's Terms

You're building a phone app called Brisbane Infinity X for healthcare gig workers in Brisbane, like physiotherapists or massage therapists at events such as Eat Street or Riverfire. The app works without internet, helping workers answer health questions (e.g., "My back hurts" → "Ice it for 15 minutes"), book appointments, predict busy times (e.g., "Hot day, expect 40% more customers"), suggest services (e.g., "Offer massages today"), post on TikTok, track supplies (e.g., "10 bandages"), save patient chats, and show a "hustle score" (like a game score, e.g., "92/100"). It has a neon-green and purple look, is easy to use, and supports five languages (English, Mandarin, Hindi, Vietnamese, Arabic). It's super fair: asks permission twice to save data, shares code for free on GitHub, works with competitors (Cliniko, Square, Jane), and donates 15% of earnings to gig worker groups. Your job is to build this in 2–3 days (by June 7, 2025, 11:59 PM AEST) using free tools, test it thoroughly, and deliver a working app with a logo and two GIFs.

1.2 Technical Terms

Develop a web-based Minimum Viable Product (MVP) using Python, Streamlit, and SQLite, hosted locally on a laptop or Raspberry Pi 4 for offline use. The app includes 20 features: health advice, appointment booking, demand prediction, service suggestions, TikTok post generation, supply tracking, conversation logging, hustle score, and a fairness dashboard. It features a neon-green (#00FF00) and purple (#800080) UI, supports translations in English, Mandarin, Hindi, Vietnamese, and Arabic, ensures two-step data consent, integrates with Cliniko/Square/Jane APIs (mock or CSV export), and publishes code to a public GitHub repository. Achieve 99% reliability with 20+ test cases, mobile-responsive design, and low-power optimization. Deliver by June 7, 2025, 11:59 PM AEST, including a Canva logo, two GIFs, and a GitHub repo link.

2 Tools and Setup

2.1 Layman's Terms

You'll need a computer (an old laptop that can run YouTube or a $50 Raspberry Pi 4 with a $10 32GB SD card). Download free software: Ubuntu (like Windows), Streamlit (makes the app look nice), SQLite (saves data like a notebook), Git (to share code), and Canva (to make a logo and moving pictures). The app must work without internet, saving everything on the computer. Use a phone to test it looks good and works fast.

2.2 Technical Terms

- **Hardware**: Laptop (min. 4GB RAM, 10GB free storage) or Raspberry Pi 4 (4GB model, 32GB SD card).
- **OS**: Ubuntu 22.04 LTS (`https://ubuntu.com`, install via USB/SD card).
- **Software**:
  - Python 3.8+ (pre-installed on Ubuntu).
  - Streamlit (`pip install streamlit`) for web UI.
  - SQLite (`pip install sqlite3`) for local database.
  - Git (`sudo apt install git`) for GitHub.
  - Canva (`https://canva.com`, free tier) for logo/GIFs.
- **Dependencies**: Install `pandas`, `numpy` (`pip install pandas numpy`), `googletrans` (`pip install googletrans==3.1.0a0`) for translations.
- **Offline Support**: Use SQLite for local storage, Streamlit caching (`@st.cache`) for offline rendering.
- **GitHub**: Create a public repository (`https://github.com`) named `brisbane-infinit`
- **Testing**: Use `pytest` (`pip install pytest`) for unit and end-to-end tests.

3 Features

3.1 1. Health Tips

3.1.1 Layman's Terms

Users type a health question (e.g., "back pain"), and the app replies (e.g., "Ice it for 15 minutes"). Show answers in English, Mandarin, Hindi, Vietnamese, or Arabic based on user choice. Ask twice before saving: "Can we save this question?" and "Confirm saving?" Show a warning: "This is general advice, not medical. Call 000 for emergencies." Test with four questions.

3.1.2 Technical Terms

- **Logic**: Hardcode a dictionary with 10 issue-advice pairs:

```
health_tips = {
    "back pain": "Ice it for 15 minutes",
    "chest pain": "Call 000 immediately",
    "headache": "Drink water and rest",
    "sore neck": "Gentle stretches for 10 minutes",
    "sprained ankle": "Rest, ice, compress, elevate",
    "muscle cramp": "Stretch and hydrate",
    "shoulder pain": "Apply heat for 15 minutes",
    "knee pain": "Rest and avoid strain",
    "fatigue": "Rest and eat a balanced meal",
    "stomach ache": "Sip water, avoid heavy food"
}
```

- **Input**: Streamlit text input (`st.text_input("Enter health issue")`) for queries.
- **Translation**: Use `googletrans` to translate responses into selected language (`st.selectbox` with options: English, Mandarin, Hindi, Vietnamese, Arabic).
- **Consent**: Two-step consent with `st.checkbox("Do you agree to save this?")` and `st.button("Confirm saving")`. Store only if both are True.
- **Output**: Display advice with `st.write`, add `st.warning("This is general advice, not medical. Call 000 for emergencies")`.
- **Database**: SQLite table `health_logs` (columns: `id INTEGER PRIMARY KEY`, `query TEXT`, `response TEXT`, `language TEXT`, `timestamp DATETIME`, `consent BOOLEAN`).
- **Test Cases**: Test four queries (e.g., "back pain", "chest pain") in two languages (English, Mandarin). Verify consent, disclaimer, and storage.

3.2 2. Appointment Booking

3.2.1 Layman's Terms

Users add appointments (e.g., "Massage, 3 PM, John"). Save them like a notebook. Add a button to save the list as a file for Cliniko or Jane. Make it easy to read on phones. Test with three bookings.

3.2.2 Technical Terms

- **Input**: Streamlit form (`st.form`) with fields: `service` (`st.text_input`), `time` (`st.datetime_input`), `client_name` (`st.text_input`).

- **Storage**: SQLite table `appointments` (columns: `id INTEGER PRIMARY KEY`, `service TEXT`, `time DATETIME`, `client_name TEXT`, `timestamp DATETIME`).
- **Output**: Display bookings in a mobile-responsive table (`st.dataframe` with CSS: `<style>table{width:100%}</style>`).
- **Integration**: Add `st.button("Export to Cliniko/Jane")` to save as CSV (`pandas.to_csv("appointments.csv")`).
- **Test Cases**: Add three bookings, export CSV, verify mobile display.

3.3 3. Busy-Time Predictions

3.3.1 Layman's Terms

Guess how busy it'll be with five rules (e.g., "34°C = 40% more customers", "Riverfire = 50% more"). Say it's a guess. Show on the main screen. Test with two scenarios.

3.3.2 Technical Terms

- **Logic**: Hardcode rules:

```
predictions = {
    "34C": "40% more customers",
    "Riverfire": "50% more customers",
    "Eat Street": "30% more customers",
    "Rainy day": "20% fewer customers",
    "Weekend": "35% more customers"
}
```

- **Input**: `st.selectbox` for conditions.
- **Output**: `st.write` with disclaimer ("Based on estimated data").
- **Storage**: SQLite table `predictions` (columns: `id INTEGER PRIMARY KEY`, `condition TEXT`, `prediction TEXT`, `timestamp DATETIME`).
- **Test Cases**: Test two conditions (e.g., "34C", "Riverfire"), verify disclaimer and storage.

3.4 4. Service Suggestions

3.4.1 Layman's Terms

If many people ask about something (e.g., "back pain"), suggest a service (e.g., "Offer massages today"). Test with two suggestions.

3.4.2 Technical Terms

- **Logic**: Query `health_logs` (`SELECT query, COUNT(*) FROM health_logs GROUP BY query`). Suggest service (e.g., "massage" for "back pain") if count > 3.
- **Output**: `st.write("Offer service today")`.
- **Test Cases**: Simulate five queries (three "back pain"), verify suggestion.

3.5 5. TikTok Post Generator

3.5.1 Layman's Terms

Create six post templates (e.g., "{service} at {name}! □ #RiverfireHealth"). Users pick one, add their name/service, and copy the text. Say it's an ad. Test two posts.

3.5.2 Technical Terms

- **Logic**: List templates:

```
tiktok_templates = [
    "{service} at {name}! □ #RiverfireHealth",
    "Get {service} with {name}! #BrisbaneHealth",
    "{name}'s {service} at Eat Street! □ #GigHealth",
    "Book {service} with {name}! #RiverfireHealth",
    "Relax with {name}'s {service}! □ #BrisbaneHealth",
    "{service} vibes with {name}! #GigHealth"
]
```

- **Input**: `st.form` with `st.selectbox` for template, `st.text_input` for `service`, `name`.
- **Output**: `st.text_area` for post, add disclaimer ("Advertisement, estimated data").
- **Storage**: SQLite table `tiktok_posts` (columns: `id INTEGER PRIMARY KEY`, `template TEXT`, `service TEXT`, `name TEXT`, `post_text TEXT`, `timestamp DATETIME`).
- **Test Cases**: Generate two posts, verify disclaimer and copy functionality.

3.6 6. Supply Tracking

3.6.1 Layman's Terms

Users list supplies (e.g., "10 bandages"). Save and show the list. Test with three items.

3.6.2 Technical Terms

- **Input**: `st.form` with `st.text_input` for item, `st.number_input` for quantity.
- **Storage**: SQLite table `supplies` (columns: `id INTEGER PRIMARY KEY`, `item TEXT`, `quantity INTEGER`, `timestamp DATETIME`).
- **Output**: `st.dataframe` for list.
- **Test Cases**: Add three items, verify display and storage.

3.7 7. Conversation Logging

3.7.1 Layman's Terms

Save patient chats (e.g., "John asked about back pain") if they agree twice. Show past chats. Test with two logs.

3.7.2 Technical Terms

- **Input**: `st.form` with `st.text_area` for chat, two-step consent (`st.checkbox`, `st.button`).
- **Storage**: SQLite table `conversations` (columns: `id INTEGER PRIMARY KEY`, `client_name TEXT`, `query TEXT`, `response TEXT`, `timestamp DATETIME`, `consent BOOLEAN`).
- **Output**: `st.dataframe` for logs if `consent = True`.
- **Test Cases**: Log two chats with consent, verify display.

3.8 8. Hustle Score

3.8.1 Layman's Terms

Show a score (0–100) based on work (e.g., 8 bookings = 80/100). Make it big and neon. Test with three updates.

3.8.2 Technical Terms

- **Logic**: Calculate: `(COUNT(appointments) * 10) + (COUNT(tiktok_posts) * 5)`, max 100.
- **Output**: `st.metric("Hustle Score", score)` with CSS `(style="color:#00FF00;f`
- **Test Cases**: Test with three bookings, two posts (score = 40), verify display.

3.9 9. Fairness Dashboard

3.9.1 Layman's Terms

Show a screen: "We're fair! We share code on GitHub, ask permission for data, give 15% to gig workers, work with Cliniko/Jane/Square." Add a feedback box. Test it works.

3.9.2 Technical Terms

- **Output**: `st.markdown` with:

  ```
  We're fair! We share code on GitHub (link), ask permission for data,
  give 15% to gig workers, work with Cliniko/Jane/Square. Complies with
  Privacy Act 1988.
  ```

- **Input**: `st.text_area` for feedback, save to SQLite `feedback` (columns: `id INTEGER PRIMARY KEY`, `feedback TEXT`, `timestamp DATETIME`).
- **Test Cases**: Verify dashboard displays, feedback saves.

3.10 10. App Look and Feel

3.10.1 Layman's Terms

Make the app neon-green and purple with two moving pictures (GIFs). Use big text, support voice readers, and add a fairness note. Ensure it works on phones. Test it looks good.

3.10.2 Technical Terms

- **UI**: Apply CSS via `st.markdown`:

  ```
  <style>
  body { background-color: #800080; color: #00FF00; }
  h1, h2, h3 { color: #00FF00; font-size: 24px; }
  button { background-color: #00FF00; color: #800080; }
  </style>
  ```

- **GIFs**: Create two GIFs in Canva (`button1.gif`, `button2.gif`), display with `st.image`.
- **Accessibility**: Add `aria-label` (e.g., `<div aria-label="Health Tips">`).
- **Responsive**: Use `st.container` for mobile layout.
- **Fairness Note**: `st.markdown("We're open, ask permission, give 15% to gig workers")`.
- **Test Cases**: Verify colors, GIFs, mobile view, voice-reader support.

4 Development Plan

4.1 Layman's Terms

Build the app in 2–3 days (June 5–7, 2025). Split work into chunks: Day 1 for health tips, bookings, TikTok posts, and UI; Day 2 for predictions, suggestions, logs, supplies, and score; Day 3 for integration, testing, and graphics. Test 20 things (e.g., questions, bookings) to make sure it works 99% of the time. Save code on GitHub, make it use little power, and work offline.

4.2 Technical Terms

- **Timeline**:
  - **Day 1 (June 5, 8 hours)**: Implement health tips, appointments, TikTok posts, UI (CSS, GIFs, accessibility). Write 8 unit tests (`pytest`).
  - **Day 2 (June 6, 8 hours)**: Implement predictions, suggestions, logs, partnerships (mock APIs), supplies, hustle score. Write 7 unit tests.
  - **Day 3 (June 7, 6 hours)**: Integrate code, run 20 end-to-end tests, optimize with `@st.cache`, create logo/GIFs in Canva, push to GitHub.
- **Testing**: Write `pytest` tests for 20 scenarios (5 health queries, 4 bookings, 3 predictions, 2 suggestions, 2 logs, 2 posts, 2 dashboard/feedback). Ensure 99% pass rate.
- **Optimization**: Use `@st.cache` for functions, limit SQLite queries to 100ms.
- **GitHub**: Commit hourly (`git commit -m "feature: health tips"`), push to `brisbane-infinity-x` repo.

5 Fairness and Compliance

5.1 Layman's Terms

Always ask permission twice to save data (pop-ups: "Can we save this?" and "Confirm?"). Show: "We comply with Australia's Privacy Act 1988." Share code on GitHub. Send data to Cliniko/Jane/Square as files. Donate 15% of earnings to Brisbane Health Co-op (log it). Make the app easy for everyone to use.

5.2 Technical Terms

- **Consent**: Two-step consent (`st.checkbox`, `st.button`) for all data storage. Set `consent BOOLEAN` in SQLite tables.
- **Compliance**: Add `st.markdown("Complies with Privacy Act 1988")` on dashboard.
- **Open Source**: Push to public GitHub repo (`git push origin main`).

- **Integration**: Mock Cliniko/Jane/Square APIs with CSV export (`pandas.to_csv`). Log 15% donation in SQLite `earnings` table (`revenue REAL, donation REAL`).
- **Accessibility**: Ensure `aria-label` on all inputs, test with screen readers.

6 Deliverables

6.1 Layman's Terms

By June 7, 2025, 11:59 PM AEST, deliver:

- Working app on a laptop that answers questions, books appointments, predicts busy times, suggests services, makes TikTok posts, tracks supplies/chats, shows a hustle score, and has a fairness dashboard.
- Logo ("Brisbane Infinity X" in neon-green/purple with a heart) and two GIFs (neon buttons).
- Code on GitHub with instructions.
- Proof 20 tests passed, works on phones, uses little power, works offline.

6.2 Technical Terms

- Streamlit app (`app.py`), SQLite database (`infinityx.db`), CSS for neon UI.
- Canva logo (`infinityx_logo.png`), GIFs (`button1.gif`, `button2.gif`).
- Public GitHub repo (`brisbane-infinity-x`) with `README.md` (setup: `pip install -r requirements.txt`, `streamlit run app.py`).
- `pytest` report showing 20/20 tests passed (`pytest test_app.py -verbose`).
- Mobile-responsive UI, optimized with `@st.cache`, offline via SQLite.

7 Troubleshooting

7.1 Layman's Terms

If something breaks:

- **Bugs**: Test 25 things instead of 20. Check error messages.
- **Slow App**: Use a faster laptop (8GB+ RAM).
- **Confused**: Re-read this document, Google errors.
- **Looks Bad**: Use Canva's pre-made neon templates.

7.2 Technical Terms

- **Bugs**: Debug with `print` or `pdb`. Add 5 extra test cases.
- **Performance**: Switch to 8GB+ RAM if Pi 4 lags. Profile with `time python app.py`.
- **Errors**: Search Stack Overflow for Streamlit/SQLite issues.
- **UI**: Use Canva's neon templates if GIFs fail.

8 Costs

8.1 Layman's Terms

You'll spend $0–$65 (laptop $0 or Pi $50 + SD card $10). All software is free.

8.2 Technical Terms

$0 (existing laptop, free tools) or $60–$65 (Pi 4, 32GB SD card). Dependencies: `streamlit`, `sqlite3`, `pandas`, `numpy`, `googletrans` (all free).

9 Final Notes

Build a fair, fun app that helps gig workers and wows investors. Follow this plan exactly, test everything, and deliver by June 7, 2025, 11:59 PM AEST. No further questions needed—just make it happen!