**SECTION 1: STAR BEHAVIORAL QUESTIONS (Answer each in 2 minutes)**

**1.1 Technical Problem-Solving**

Tell me about a time you faced a really hard technical problem that wasn't straightforward. Walk me through what happened, what you did, and how it turned out.

**1.2 Failure & Learning**

Tell me about a time you failed. What happened, and what did you learn from it?

**1.3 Disagreement & Collaboration**

Tell me about a time you disagreed with a team member or manager. How did you handle it?

**1.4 Initiative & Leadership**

Tell me about a time you took initiative and did something without being asked.

**1.5 Asking for Help**

Tell me about a time you got stuck and had to ask for help. How did you handle it?

**1.6 Handling Unexpected Challenges**

Tell me about a time something unexpected happened during a project. How did you adapt?

**1.7 Working in a Great Team**

Tell me about a time you worked in a team that was really great. What made it great?

**1.8 Difficulty in a Team**

Tell me about a time working in a team was difficult. What happened and how did you resolve it?

**1.9 Going Above & Beyond**

Tell me about a time you went above and beyond to deliver something excellent.

**1.10 Handling Client/Manager Being Wrong**

Tell me about a time a client or manager was wrong about something. How did you handle it?

**1.11 Goals Under Pressure**

Tell me about a time you set a goal but couldn't meet it. How did you handle it?

**1.12 Greatest Achievement**

What's your greatest achievement? Why is it meaningful to you?

**SECTION 2: LIBRARIES & FRAMEWORKS (Technical Depth)**

**2.1 Flask**

* Why Flask over Django?
* How do you structure a Flask app at scale?
* How do you handle errors and exceptions?
* What's the difference between development and production deployment?
* Explain request context vs app context.
* How would you deploy Flask to production? (Gunicorn, uWSGI, etc.)

**2.2 python-telegram-bot**

* Explain polling vs webhooks for Telegram bots. When use each?
* How do you handle long-running operations in a bot without blocking?
* How do you secure your bot token? (Never hardcode, right?)
* How do you store user state across interactions?
* What are handlers, filters, and context in telegram-bot?

**2.3 Twilio**

* What are rate limits for Twilio API?
* How would you handle Twilio API failures gracefully?
* What's the cost model for Twilio? How do you budget for it?
* Explain sync vs async Twilio API calls—when use each?
* What can you do with Twilio besides phone lookups?

**2.4 OpenAI API**

* How do you manage costs with OpenAI API?
* What's the difference between temperature and max\_tokens parameters?
* How do you prevent hallucinations from GPT?
* Explain few-shot learning with GPT. Give an example.
* What's token counting? Why does it matter?

**2.5 Oracle Database**

* How does indexing work in Oracle? Why do indexes improve performance?
* What's the difference between primary key and foreign key?
* Explain transactions and ACID properties.
* How do you optimize slow SQL queries? (EXPLAIN PLAN?)
* What's partitioning vs sharding?
* How do you handle concurrent writes to avoid conflicts?

**2.6 Pandas & NumPy**

* What's the difference between loc and iloc in Pandas?
* How do you handle missing data in a DataFrame?
* Explain groupby and aggregation with an example.
* What's the difference between merge and join?
* When would you use apply() on a DataFrame?
* Explain broadcasting in NumPy.

**2.7 Scikit-learn**

* Explain train/test split. Why 80-20?
* What's cross-validation and why use it?
* How do you tune hyperparameters? (Grid search vs random search?)
* Random Forest vs Decision Tree—when use each?
* What's overfitting and how do you detect it?
* Explain precision, recall, F1 score, and accuracy.

**2.8 Keras/TensorFlow**

* What's the difference between Keras and TensorFlow?
* What's dropout and why use it?
* Explain backpropagation briefly.
* What's activation function? Why use ReLU?
* How do you prevent overfitting in neural networks?

**2.9 PyTorch**

* What's autograd in PyTorch?
* Difference between PyTorch and TensorFlow?
* What's a computational graph?
* How do you train a model in PyTorch? (optimizer, loss, backward pass)

**2.10 NLTK (Natural Language Processing)**

* What's the difference between tokenization, stemming, and lemmatization?
* What are stopwords? When do you remove them?
* Explain POS tagging (part-of-speech tagging).
* How would you build an NLP pipeline with NLTK?

**2.11 Hugging Face Transformers**

* What's BERT and how is it different from traditional ML models?
* Explain RoBERTa vs BERT. Which is better and why?
* What's transfer learning in NLP?
* How do you fine-tune a transformer model on your own data?
* Explain what a tokenizer does in Hugging Face.
* What's the attention mechanism?

**2.12 Gensim**

* What's Word2Vec and what problem does it solve?
* What's LDA (Latent Dirichlet Allocation)?
* When would you use Gensim over other NLP libraries?

**SECTION 3: PROJECTS (Deep Technical Questions)**

**3.1 Phone Checker Bot**

**Architecture:**

* Walk me through the Phone Checker Bot from start to finish.
* What problem does it solve?
* What's the architecture? How do the pieces fit together?
* Why integrate Twilio, DuckDuckGo, and OpenAI specifically?

**Performance:**

* You mention 40% improvement with caching. What was the baseline?
* How did you measure the 40% improvement? (Methodology?)
* Why Oracle Database for caching instead of Redis or Memcached?

**Scalability:**

* How would you scale this to 1 million concurrent users?
* What breaks at scale?
* Database strategy? (Sharding? Partitioning?)
* How would you handle API rate limits?

**Reliability:**

* What if Twilio API goes down?
* What if DuckDuckGo Search fails?
* How do you handle false positives? (Legitimate number marked as spam?)
* What's your fallback strategy?

**Cost & Deployment:**

* What's the monthly cost of running this bot?
* How is it deployed? (Docker? Kubernetes?)
* What's the latency per check?

**Improvements:**

* Is this deployed to real users? How many?
* What would you do differently if rebuilding it today?

**3.2 Anti-Money Laundering (AML) Detection System**

**Problem & Approach:**

* Walk me through the AML system you built.
* What was the business problem?
* Why 94% accuracy? What does that actually mean?

**Metrics & Evaluation:**

* 94% accuracy alone isn't enough for fraud. What about precision and recall?
* Which matters more—precision or recall for fraud detection and why?
* How did you handle class imbalance? (Fraud is ~1% of data)

**Technical Details:**

* What features did you engineer? How many?
* What algorithm did you use? Why that one?
* How did you prevent data leakage?
* How did you validate your model?

**Team & Contribution:**

* You say 5-member team. What was YOUR specific contribution vs others?
* What was the hardest part for you specifically?

**Production:**

* Was this deployed in production?
* How do you monitor for model drift?
* What happens when accuracy degrades?

**3.3 AI Plagiarism Detection**

**Problem & Solution:**

* Tell me about the plagiarism detection project.
* What problem does it solve?
* Why build it? (Real use case?)

**Model Selection:**

* You chose RoBERTa. Why not BERT or GPT?
* How did you benchmark? (Methodology?)
* What were the trade-offs?

**Accuracy & Limitations:**

* You say 92% accuracy. What about false positives and false negatives?
* Can it detect paraphrased AI text?
* Does it work on different languages?
* Edge cases: code, technical writing?

**Implementation:**

* Did you fine-tune RoBERTa or use it pre-trained?
* How long does inference take per essay?
* What's your data pipeline look like?

**Deployment:**

* Is it deployed? Real users?
* Inference latency requirements?

**Future-proofing:**

* ChatGPT keeps evolving. Your model becomes outdated. How do you handle that?
* Would you open-source it?

**3.4 Inverse Cooking (Recipe Generation from Images)**

**Problem & Architecture:**

* Walk me through the Inverse Cooking project.
* How does image-to-recipe generation work?
* What's the architecture? (CNN + NLP?)

**Technical Decisions:**

* What CNN architecture did you use? (ResNet, VGG, Custom?)
* Did you use transfer learning?
* How does T5 generate recipes? (Sequence-to-sequence?)
* Why combine vision + NLP?

**Performance:**

* You mention 85% accuracy. How did you measure it?
* What does 85% accuracy mean for this use case?
* Inference time?

**Edge Cases:**

* What if image has multiple dishes?
* What if image quality is poor?
* Does it work for all cuisines?

**SECTION 4: CLOSING QUESTIONS**

**4.1 Motivation & Interest**

* Why are you interested in ANZ specifically? (Not just FAANG in general)
* What space in banking interests you most?
* Why now? (You graduate May 2026—what's driving this?)

**4.2 Long-term Thinking**

* Where do you see yourself in 5 years?
* What do you want to become expert in?
* Do you see yourself more as specialist or generalist?

**4.3 Growth & Learning**

* What's one skill you want to develop in the next 2 years?
* How do you stay current with technology?
* Tell me about something new you learned in the last 3 months.

**4.4 Challenges & Fit**

* What do you find most challenging about ML/Data Science?
* What's NOT your strength?
* Why would you succeed in a fast-paced, high-pressure environment?

**4.5 Compensation & Logistics**

* What's your salary expectation? (If asked)
* Are you willing to relocate? (Melbourne to Sydney, etc.)
* When would you be available to start?

**4.6 Your Questions**

* What questions do YOU have for us?

**NOTES ON TIMING**

**STAR Questions:** ~2 minutes each (total 24 minutes for all 12) **Libraries:** ~1 minute per question (asked selectively, not all) **Projects:** ~3-5 minutes depending on depth **Closing:** ~1 minute each

**Total interview time:** 45-60 minutes