* Data flow diagrams.
* Textbook ‘documentation’ section.
* Follow structures from the textbook.
* Go over ISYS unit resources.
* Quantitative metric for data privacy; questionnaires, surveys, something that provides an answer.
* Ultimate question: is system safe? How can you measure and prove this?
  + Simulations, tests.
* Rewrite introduction:
  + Be more ‘research’-focused.
  + Focus on quantitative results from different options of system designs; “system A is better than system B because of these differences”.
* Design tests before building system; break system then build around these vulnerabilities.
* Testing that:
  + System behaves as it should given compliant user.
  + System behaves as it should given non-compliant user.
    - Lock them out
  + Malicious user, behaves as it should.
* Why not use another system?
  + Expenses
  + Doesn’t meet user requirements
  + System abilities:
    - Scalability
    - Flexibility
* ER diagrams, data flow diagrams.
* How to design a system like this? What steps to follow and then how to test? Best configuration for the databases?
* User requirements 🡪 design (3 different designs) 🡪 test plan 🡪 prelim work (e.g. using matlab to keep stuff in a database check r) 🡪 Thesis B plan: run other 2 designs with test plan 🡪 plan over the 15 weeks of next semester, how’s it gonna all work and be tested
* If the 3 designs don’t’ work or have flaws propose a 4th that combines the best of all
* HD: include preliminary test plan, show them a shitload of work.
* Implement something!!!
* 5-10 hours writing thesis, 5-10 hours practicing with matlab/practical work; create mock databases, fields, stuff you can test with. Tests: making sure data is going in the right places.
  + Test plan: these are the queries to retrieve data, metrics for safe vs unsafe.
* Break name into initials? Perhaps?
* SQL-lite.
* Second marker for this thesis? Ask around.
  + Security
  + Data privacy
  + Databases
  + Pure profiles for MQU staff; it’s their record of research. Researchers.edu.mq or something like that