**Title page**

* Table of Contents
* Glossary

**Chapter 1. Introduction**

* Problem statement
* use material from your proposal outline

**Chapter 2. Method**

* identify suitable tools and methodology (don’t just say agile)
* identify risks
* include ethics if needed.
* version/document control strategy

**Chapter 3. Research**

* primary research - data you collected yourself
* secondary research - relevant literature/white-papers/standards
* informal sources - user stories (epics)
* case-studies
* technology selection where required - pugh matrix
* Main argument and alternative arguments

**Chapter 4. Requirements**

* Capture user-stories / use-cases (and use-case diagrams)
* Derive (suitably numbered e.g. FR1, FR2,...) functional requirements (prioritised).
* Use Quality assurance (ISO/IEC 9126 Software Engineering: Product quality) to develop Non-Functional Requirements  
  e.g. numbered NFRs (e.g. NFR1, NFR2, ...) organised by quality:

      1. Usability  
      2. Functionality  
      3. Reliability  
      4. Portability  
      5. Efficiency  
      6. Maintainability

* For a design process: mood board & persona

**Chapter 5. Design**

**\*\*\* Don't forget to tell us what your project actually does \*\*\***

* Software architecture
* Relevant UML diagrams (e.g. component diagrams, sequence diagrams)
* Code snippets to highlight core algorithms, or pseudo-code.
* Wireframes for screen design
* UX design map

**Chapter 6. Results**

* Screenshots to demonstrate your use-case scenarios
* Automated or Manual Testing, Requirements Traceability Matrix

**Chapter 7. Conclusion and next steps**

* Include critical reflection on the project content and process - e.g. Gibbs'

**References**

* (UWE) Harvard

**Appendices**

* questionnare consent form
* participant information sheet
* questionnare questions
* do **not** include personal data