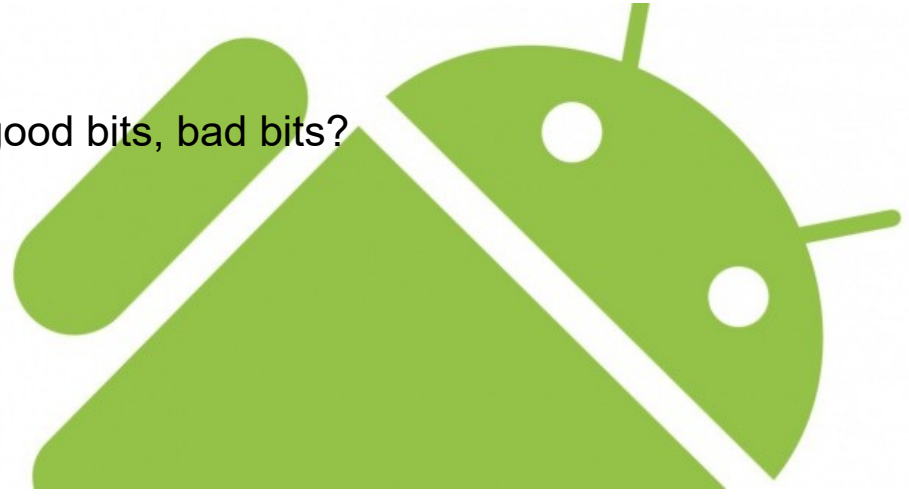
Android Projects Group

Group Meeting 4:

Writing your Final Report

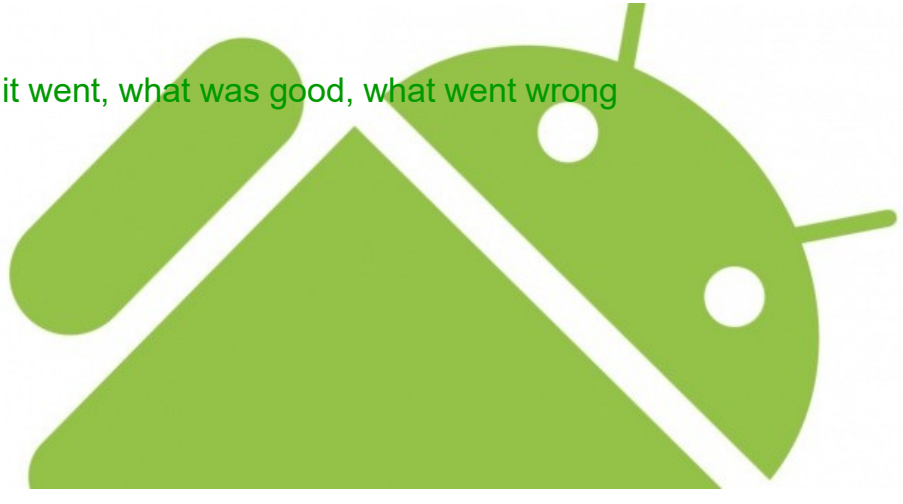
Writing your Final Report

- Your Final Report is the main outcome of your Project
- Your Final Report shows everything you have achieved
- It is a big piece of work!
- There is no fixed way to write it
- You can write it in a way that best expresses how you feel about your work
- Your report should read a bit like a long magazine article:
  - Introduce the reader to your project idea
  - Tell a story of what happened
  - Show from start to finish what you did
  - Explaining what you did and how it works
  - Ending with a critical review - how did it go, good bits, bad bits?



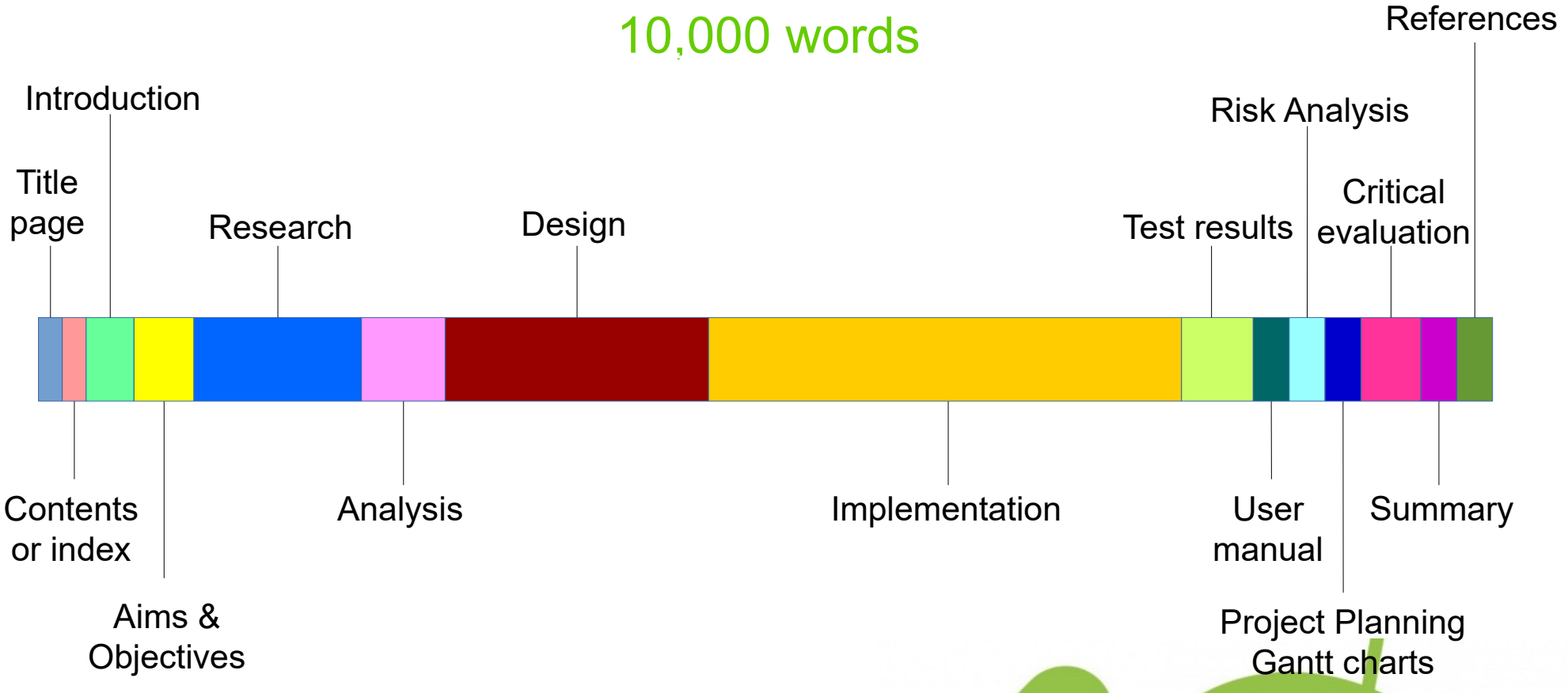
Report Headings

- **Title page**
- **Index**
- **Introduction:** 1 page on your project idea and what it's about
- **Aims and Objectives:** 1-2 pages on what you wanted your project to do
- **Research:** 6-12 pages on the findings of your literature review
- **Analysis:** 4-6 pages based on your literature review conclusions
- **Design:** 8-16 pages showing your design from sketches to diagrams
- **Implementation:** 10-20 pages showing how you wrote your code
- **Test results:** 4-8 pages showing unit test results
- **User Manual:** 1-3 pages showing your user manual
- **Risk Analysis:** 1-2 pages on GitHub and risk management
- **Project Planning:** 1-2 pages on updating Gantt charts
- **Critical Evaluation:** 2-4 pages talking about how well it went, what was good, what went wrong
- **Summary:** 1-2 pages wrapping it up
- **References**



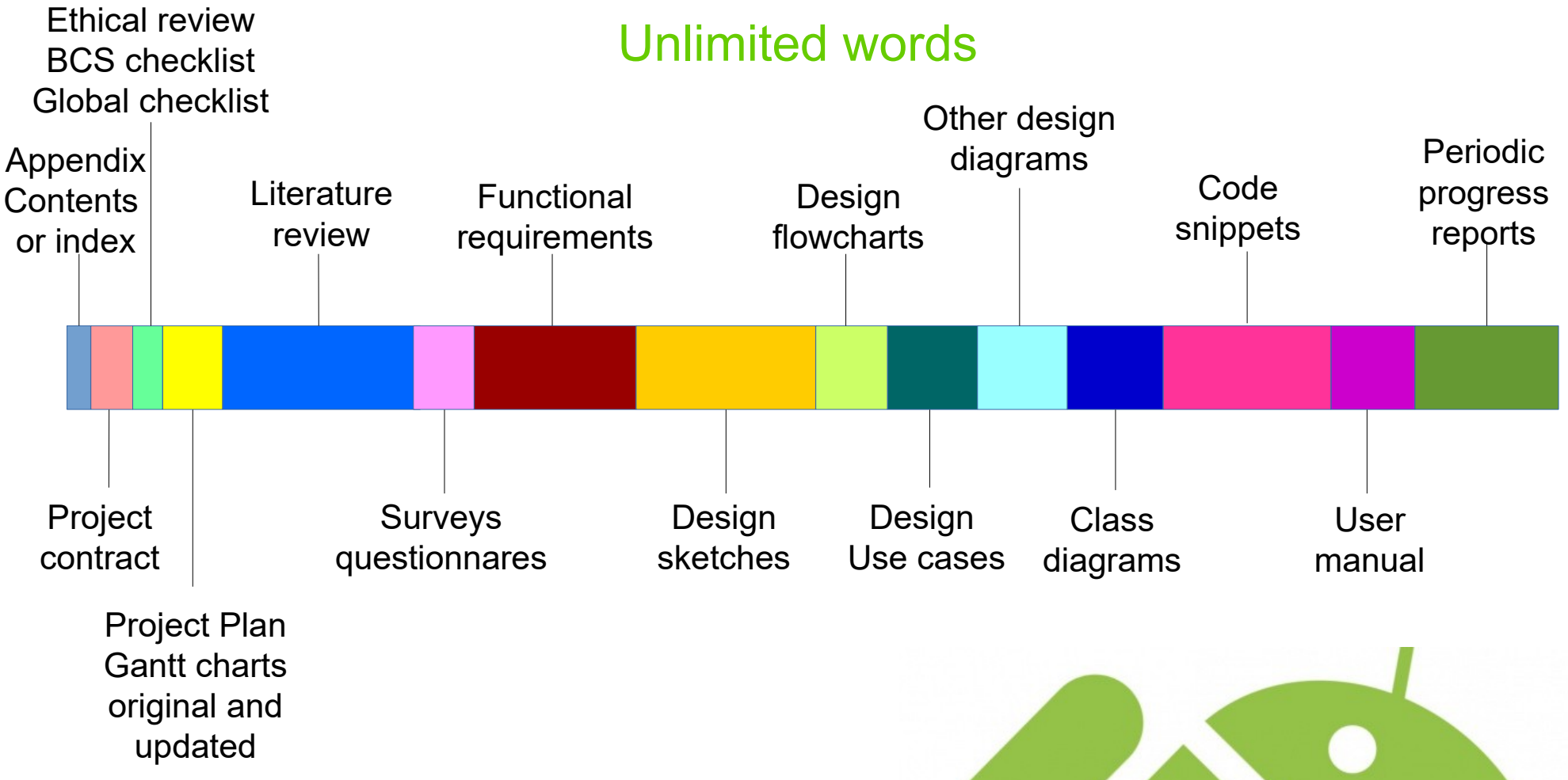
Main Report Structure

10,000 words



Appendix Structure

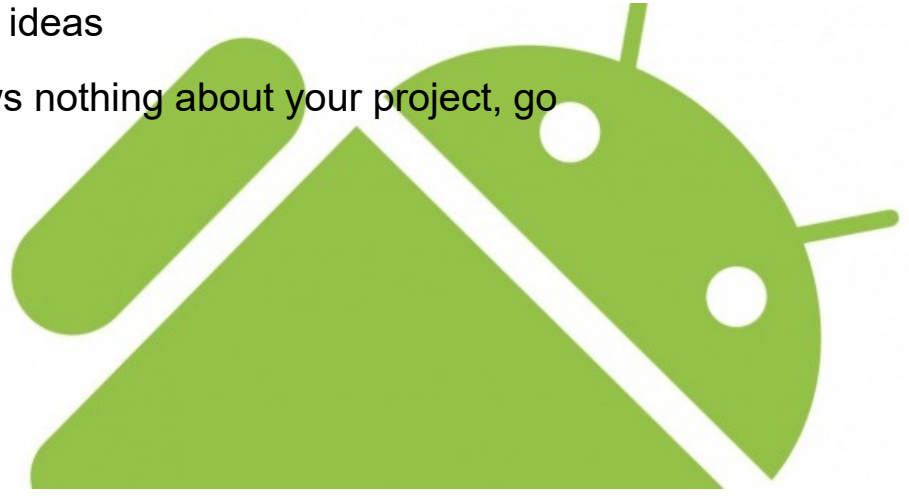
Unlimited words



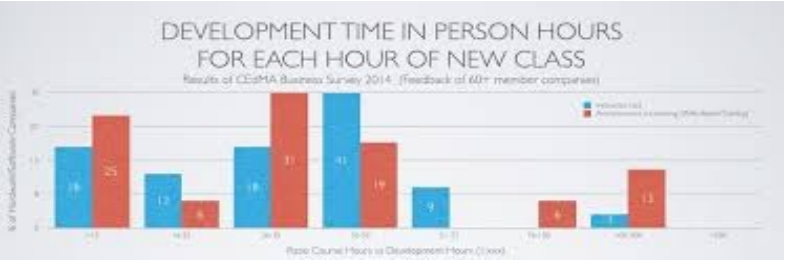
Report Contents

- **Introduction**
- Around 200-600 words, sentences rather than bullet points.
- Your motivation
- Use the Third person - ‘The work was done’, ‘A project was proposed’...
- Imagine somebody is reading your report and knows nothing about your project, give them an idea what it’s about

- **Aims & Objectives**
- Around 1-2 pages, more if you need
- The aim of this project was to…
- Look at you Project Contract and Interim Report for ideas
- Imagine somebody is reading your report and knows nothing about your project, go through each key aim or objective of the project



Report Contents



- **Research**
- This is talking about your literature review
- 6 to 12 pages, more if you need
- Use References if you quote something, and put the in the References section later in the report
- If your had a few summary tables, then use them
- If your had a few survey results tables, then use them
- You can also rewrite your original literature review if you like
- Talk about your findings from your literature review
- This is from your first report literature review, put that in the Appendix, talk about it here
- Imagine somebody is reading your report and knows nothing about your research, or what you have researched about your project area, give them an idea what you have learnt.

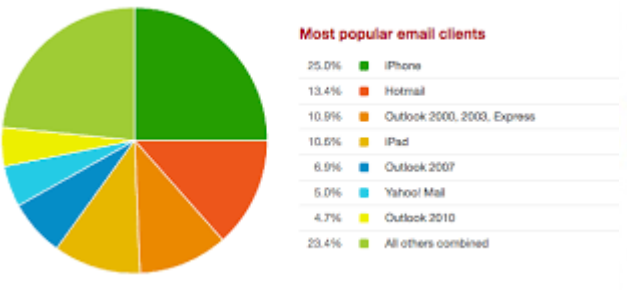
Pecentage of responders that frequently play or are addicted to:		
		Response Percent
Purchased Action Games		37%
All Other Purchased Games		30%
Pay to Play Internet Games		24%
Free Internet Games		15%
Window Games (Free Cell, Solitaire, etc)		12%

Report Contents

- **Analysis**
- Summary of your research and literature review
- What does the research mean
- How does the research influence how your app works
- What are the results of your research
- 4 to 6 pages, more if you need
- This research is from your first report literature review conclusions, put that in the Appendix, talk about it and expand it here
- You can also rewrite your original literature review conclusions if you like
- Imagine somebody is reading your report and knows nothing about what your research means, so you have found things out but what does it mean?

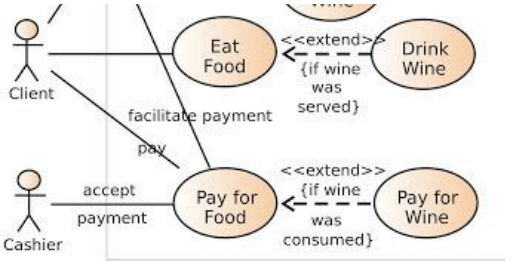
Analyzing table data

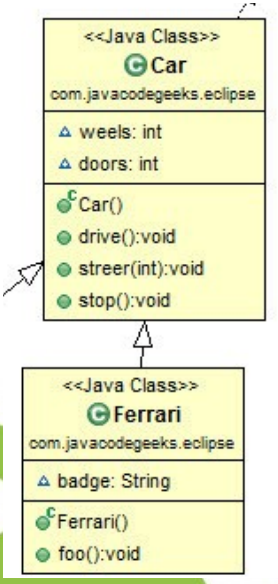
level	1998	1999	2000	2001	2002	2003	2004	2005
advanced	7	9	15	18	20	24	29	35
proficient	17	15	18	27	24	27	28	27
needs improvement	24	23	22	30	31	29	28	24
failing	52	53	45	25	25	20	15	15

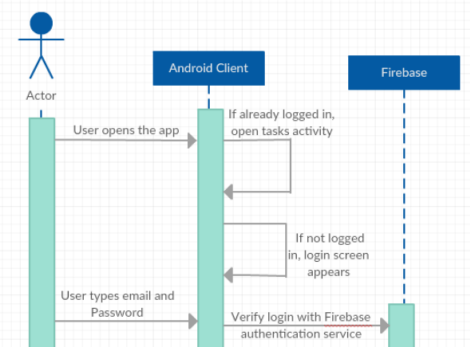


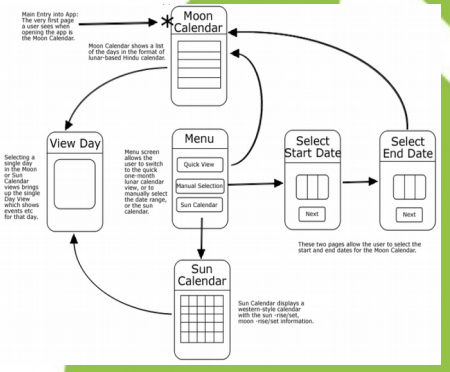
Report Contents

- **Design**
- How does the research and analysis lead to your app design
- App design sketches
- Design diagram snippets
- Put the full diagrams in the Appendix
- 8 to 16 pages, more if you need
- Typical examples include: Class Diagrams, Use Cases, Sequence Diagrams, anything that showcases the design of your system
- These designs will be placed in the appendices with snippets shown here
- Imagine somebody is reading your report and knows nothing about your app design, explain how you designed your app to them



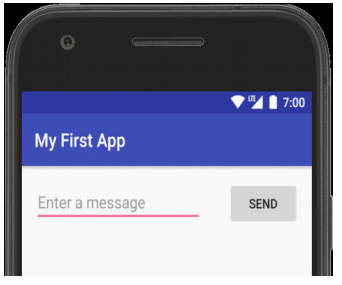


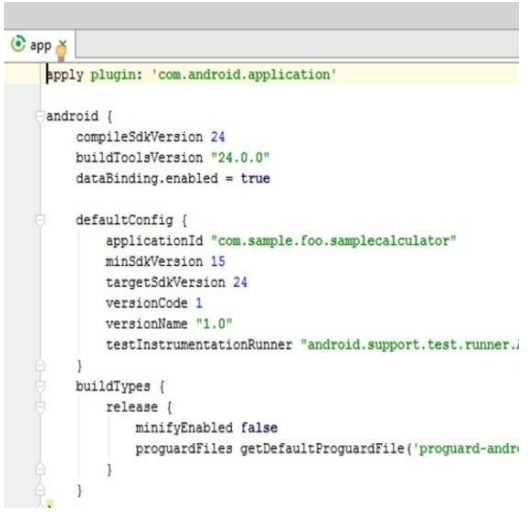


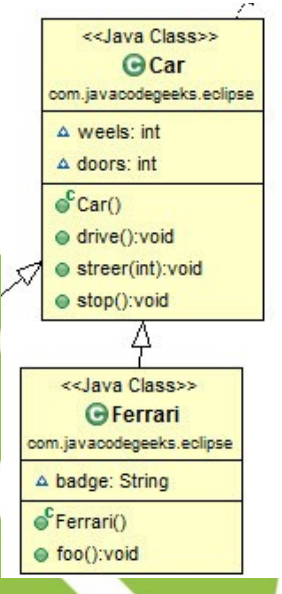


Report Contents

- **Implementation**
- Describe how you wrote your code
- This can be a long section, 10 to 20 pages
- Break it down into modules and describe them
- For each module show code snippets, class diagrams, etc
- Refer to fulfilling Functional Requirements
- A step by step account of how you developed your software.
- You can include fragments of code and screenshots, refer to code snippets in the appendix
- Your appendices will contain screenshots for your entire application that can be referred to.
- Imagine somebody is reading your report and knows nothing about how you wrote your code, explain it here







Requirement ID	Description	Functional requirements	Non-functional Requirements
1	capture image	invoke camera display image respond to capture button capture image	Low latency, <100ms for capture
2		save image to bitmap	none

Report Contents

- **Test results**
- Test results for the app, tests based on functional requirements
- Test tables in the appendix, snippets here
- Think of a test for each Functional Requirement
- Test all of the things your app does, text boxes, buttons etc
- Put the test results in the Appendix
- Show snippets from the test results here

Test task	Functional Requirement	Input	Actual Result	Pass/Fail	Comments
Show camera image	Requirement 6.1 Show camera image	Camera button selected	Camera image displayed. Capture button displayed	Pass	Buttons may need different layout to be easier to select
Capture image	Requirement 6.2 Image capture	Capture button selected	Camera captures image. Saves to bitmap.	Pass	
Identify image	Requirement 6.3 Identify image	Image from camera	Butterfly name	Pass	Name retrieved from google server

Report Contents

- **User manual**
- Write a short user manual
- Use screen shots
- Put it in the Appendix
- Mention it here

Capturing an Image

1. Press the capture button on the home screen
2. The capture a screen is shown you will see a camera image
3. Point the camera at the butterfly
4. Let the camera focus
5. Press the capture button
6. The captured image will be displayed

Identification

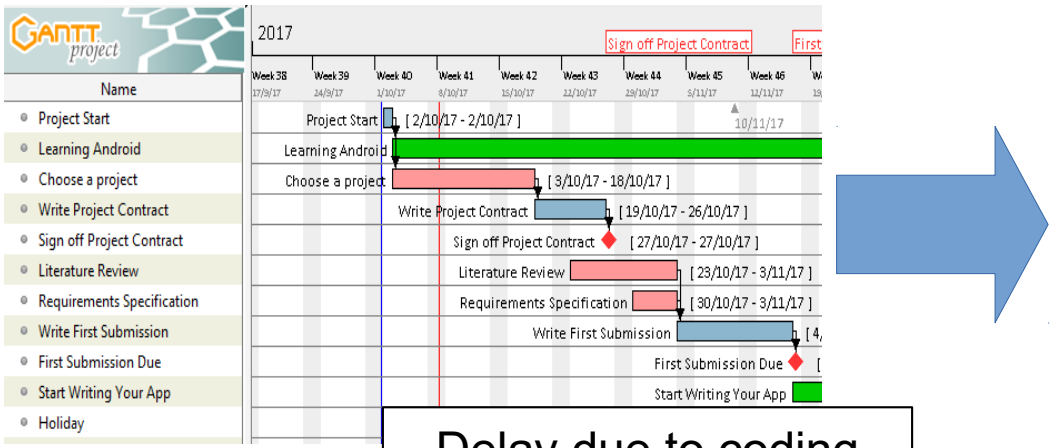
1. Once an image is captured you can identify it or try again
2. to try again press the capture button again
3. to identify press the Identify button

Report Contents

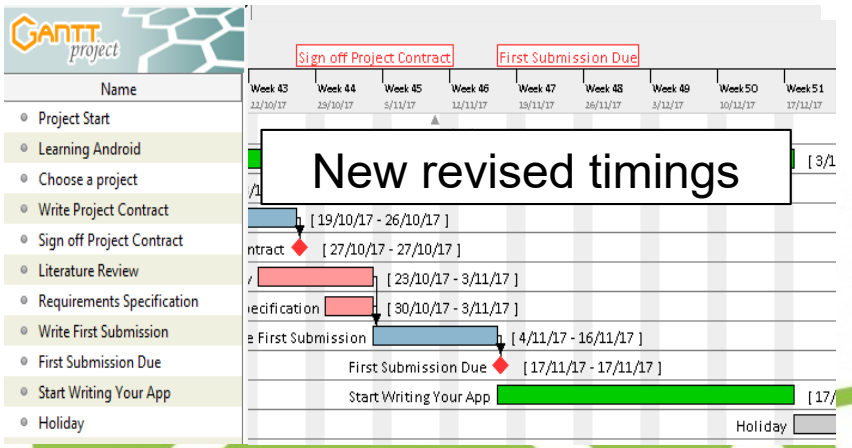
- **Risk Analysis**
- Did you risk mitigation measures work
- Did you lose data
- Did GitHub work correctly
- Did your backups work
- Use the risk analysis from your Project Contract

Report Contents

- **Project planning**
- Did your time risk mitigation measures work
- Did you implement all the core functions, or leave some out due to itme?
- Did your time management work - got ill, got stuck on development?
- Update your original Gantt to show how work changed
- Put them in the Appendix, and refer to them, show how plans changed
- You get marks for showing you managed changes!

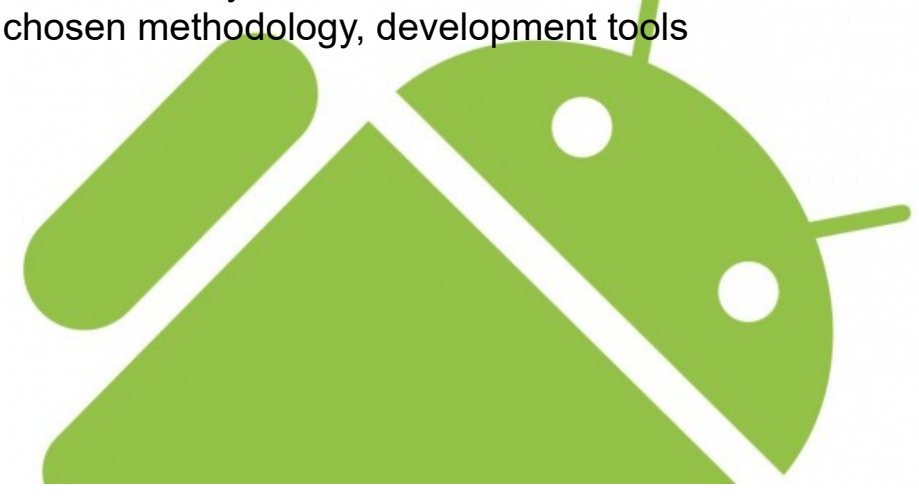


Delay due to coding issue



Report Contents

- **Critical evaluation**
- Use the first person - ‘I thought the app worked well’, ‘I had problems with...’
- A talk and summary about how the whole project progressed
- Talk about how you overcame problems
- Talk about fulfilling the Project Contract
- 2-4 pages
- There are different ways to structure this but one approach is to base it on the objectives in your project contract.
- You should write about what is both good and bad about your software (and project experience more generally) – try and focus on the positives, but clearly do not neglect the obvious negatives or enhancements.
- Obvious areas are your software, research, requirements, etc. But you can also consider how you managed the project, how the plans evolved, the chosen methodology, development tools you used, etc.
- Also consider future enhancements to your software.

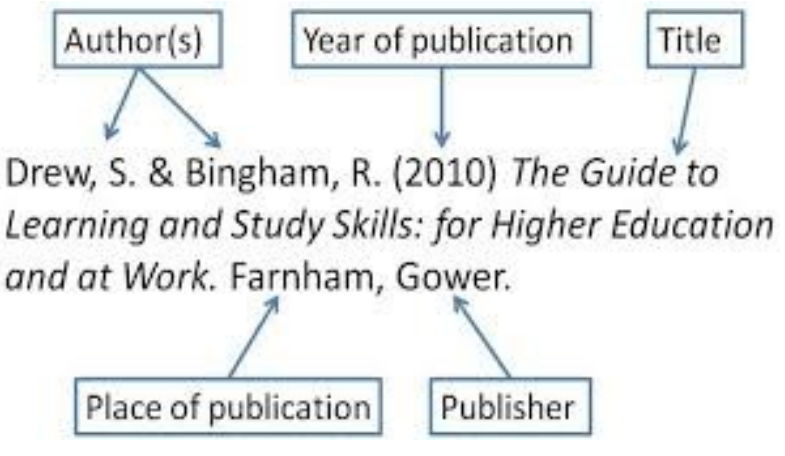


Report Contents

- **Summary**
- Use the first person - ‘I felt it went well’, ‘I learnt a lot about’...
- Wrap it up.
- How did it go?
- Were you happy?
- 200-600 words
- Sound up-beat! Positive!
- What will you do next with the app!

Report Contents

- **References**
- Use Harvard notation, like your first report
- Try to have some references - you get marks for them!
- Do the references properly.



Appendix Contents

- **Appendix - unlimited size**
  - Project Contract
  - Ethical Review, BCS Checklist, Global Checklist
  - Project Plan - Gantt Chart and other project planning presentations **- show original and updated charts**
  - Literature Review **- you can rewrite your old one**
  - Any survey or questionnaire result
  - Functional Requirements **- you can rewrite your old one**
  - Design Sketches
  - Design flowcharts
  - Design Use Cases
  - Design Class Diagrams
  - Design Data Structures
  - Design ERD's
  - Design Sequence Diagrams
  - Any other design diagrams
  - Code listings - **snippets, not all of it, with explanation>**
  - Brief user manual
  - Periodic Progress Reports **- don't forget, you get marks for these - order is oldest first**

The Appendix is where you put all the big stuff that won't fit into your main report!

Keeping stuff in the Appendix makes your main report a better read - refer to the Appendix from the main report

Marking Grids 1/4

Main report		Clear fail <30%	Marginal fail 30-39%	Bare pass 40-49%	Clear pass 50-59%	Very good 60-69%	Excellent 70-79%	Excep 80-(
Presentation	Acknowledgements, length, spelling, grammar, written style, table of contents, page numbers referencing etc.	< Make it look nice!						
Background and justification	Clear discussion and justification of why the project is useful, necessary and/or important to be undertaken	< Introduction, Aims & Objectives						
Discussion of evolution of main components	Identification of significant use cases and discussion how they evolved from inception to implementation	< Design						
Application of theory	Clear links to existing theory and evidence of how extant knowledge is applied while developing the project, in the design of the final product etc.	< Analysis and Design						
Critical analysis	What went wrong and what was right? What could be done differently next time?	< Critical evaluation						
Risk assessment	Relevant risks of project failure have been identified and properly mitigated. Risk analysis supplied	< Risk analysis						
System Design	There is good quality system design documentation (using recognised notation as agreed with the supervisor)	< Design, and the Appendices						
Testing	Strategies for testing the main aspects of the systems have been presented.	< Test planning and results						
Referencing	Proper referencing, according to DMU guidelines, clearly distinguishing own work from others	< References - do a few to get marks!						

Marking Grids 2/4

Appendices		Clear fail <30%	Marginal fail 30-39%	Bare pass 40-49%	Clear pass 50-59%	Very good 60-69%
Software development strategy	Clear evidence of an appropriate overall strategy e.g. XP/SCRUM/SSADM etc.	< Design, Implementation				
Appropriateness of documentation	Sensible selection of documentary artefacts which appropriately inform the design of your system	< Write something for each section				
Testing	Suitable testing at all levels of the system e.g. code & usability	< Implementation unit tests, Test results				
Management	Clear evidence of regular meetings with supervisor and	< Periodic Progress Reports!				

Marking Grids 3/4

Main report		Clear fail <30%	Marginal fail 30-39%	Bare pass 40-49%	Clear pass 50-59%	Very goo 60-69%
documentation	indication of content discussed actions taken	< Evidence that you reacted to changes				
Analysis and design documentation	Selected documentation makes use of correct notation and is logically correct	< Design uses correct diagrams				
Alignment of documentation and implementation	Documentation matches the system created e.g. ERD aligns with tables etc.	< Design and Implementation match				
Project Management		Clear fail <30%	Marginal fail 30-39%	Bare pass 40-49%	Clear pass 50-59%	Very goo 60-69%
Management meetings	Meaningful and consistent engagement with your supervisor	< Periodic Progress Reports!				
Response to change	Clear demonstration of ability to respond to changes in circumstances	< Project Planning, revised Gantt charts				
Project planning	The student was able to proactively plan and manage the project, replanning as necessary	< Project Planning, revised Gantt charts				

Marking Grids 4/4

Viva		Clear fail <30%	Marginal fail 30-39%	Bare pass 40-49%	Clear pass 50-59%	Very good 60-69%	Excellent 70-79%
Professionalism and appearance	The student looks and acts like a computing professional	< Keep calm, be cool, look good					
Timing and delivery	Demonstration filled the available time appropriately with time for questions	< Practice your viva at home first					
System coverage	All important use cases covered	< Make sure core functions work					
Question handling	The student is able to defend the system	< Make sure you know how it works					
The System		Clear fail <30%	Marginal fail 30-39%	Bare pass 40-49%	Clear pass 50-59%	Very good 60-69%	Excellent 70-79%
Quantity of work	Appropriate for final year project.	< Try to write something for all the sections					
Product	The product meets the objectives of the agreed project contract (completeness)	< Make sure app meets Project Contract core functions					
Usability	The project demonstrates good usability	< Big buttons, nice interface, easy to use					

Main report		Clear fail <30%	Marginal fail 30-39%	Bare pass 40-49%	Clear pass 50-59%	Very good 60-69%	Excellent 70-79%
Robustness	Handles user errors in a robust and informative manner	< App has no bugs, handles incorrect user input					
Sound design and implementation	Appropriate for the technologies in use	< Show that Design and App uses latest technology					

Summary

- Finish or stop development and coding by start of Easter holiday
  - Try to get core functions working
  - Don't worry about other functions
- Start writing your report over the holidays - 3 weeks work
  - Your Report is more important than your App - more marks!
  - Remember the Marking Grid when writing your report
  - Try to write something for each of the sections
- Finish first draft of your report by first week back and show me:
  - Feedback
  - Hints and tips
- Deadline - Friday 20th April - TurnItIn - single PDF file
- Vivas - to be arranged - possible group meeting...