# About the report

Via this report you will communicate with us various aspects of SPA i.e., design, implementation, testing and project management. One of the most important skills that we want you to develop in this module is written communication. You should provide all relevant information in an accurate, succinct and effective manner. Remember, **quality is more important than quantity**. If you can convey the message via one well-written UML diagram, it is preferred over two pages of descriptions. Similarly, if you can communicate the project plan visually (e.g., using Gantt charts), it is preferred over tables and descriptions. Roughly the report should contain the sections specified in this template. We will try to specify what exactly should each (sub)section of the report should contain.

Also remember: this is a report, and not a bulleted laundry list of things done in the project. Just because you are required to be succinct doesn’t mean you write a bullet list.

**The report is meant to be cumulative**; so be judicious on how you want to fill it up. Unlike some of the reports we have seen in the past, you should not discard the Iteration 1 deliverable and start afresh for Iteration 2 and subsequently Iteration 3. Rather, your report should describe the entire project including all the iterations.

It is OK to be detailed for iteration 1, provided, you distil it appropriately in the future iterations. Remember, there is a page length restriction (as specified in the Format next).

## Formatting and content guidelines

* The report should be written in 12-point font, with 1-inch (~2.5 cm) margin on all sides.
* Line spacing: 1.15 (in MS Word; use equivalent values if you are using other tools like PageMaker or LaTeX).
* Section heading/subheadings should be formatted using Headings1, 2 and 3 respectively. If any section has more than 3 levels of subsubsection nesting, probably it needs to be reworked.
* Overall page length restrictions: Maximum 120 pages in the final report (which means iteration 1 and 2 should be appropriately scaled, OR, in iteration 3 you need to trim earlier portions appropriately). You can have unlimited Appendices. The teaching team is not obliged to read the Appendices while grading. So include all important/relevant information within 120 pages.
* The report should have 4 (5 including appendix) main parts
  + Abstract
  + Part1: Technical report
  + Part2: Project management
  + Part3: Conclusion
  + Appendix
* The following are strictly forbidden:
  + Writing stories where things can be explained using a reasonable visual aid or a short paragraph.
  + Excessive screenshots (keep screenshots to minimal) and sample code snippets – reason: they are hard to maintain
  + C++ code snippets and method headers (in the name of API); use abstract API instead, where necessary
  + Try to use minimal sample SIMPLE programs. Use them as running examples; it helps to maintain context and continuity to the reader.
    - It also means, you need to think of and come up with one or two SIMPLE programs that can highlight the design, implementation and advantages/disadvantages of your implementation of all aspects of your SPA.
  + Unnecessary explanations (e.g., a sequence diagram should be well written and self-explanatory; you should not include separate step-by-step description of the interaction – i.e., what can be easily observed from the diagram).
    - However, you need to include a high-level overview of the diagram. You cannot simply dump diagrams and complete the section.
  + Giving examples of obvious things (e.g., when you say a list, the teaching team understands what a list is, you don’t have to give additional examples) when explaining data structures or algorithms used in your implementation.
  + Using tables to compare, and explaining each cell in the table. Rather, you are encouraged to write, despite the shortcomings, if any, why did you choose something over the other among the items you compared.
  + Don’t repeat information in the tables; if something is similar, explain only one and highlight differences (e.g., when you are talking about different relationships).

The actual report template follows.



CS3203 Software Engineering Project

AY21/22 Semester 1

Project Report

Team \_\_

|  |  |  |
| --- | --- | --- |
| Team Members | Student No. | Email |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

**Consultation Hours:** <Insert day and time here>

**Tutor(s):** <Insert name here>

Abstract

SPA comes with a pre-defined set of requirements. You are expected to implement features that satisfy almost all the given requirements. In this section, you are expected to write anything unique you did that differentiates your project. Treat this as a pitch that you can use to sell your project to potential recruiters.

Some examples (non-exhaustive):

1. Any unique algorithm or data structure you implemented in SPA
2. Any specific optimization you performed in the data store (PKB) or the query parser (PQL)
3. You can briefly describe your extension and how it benefits SPA users
4. Any significant deviation of your architecture design from the given one can be highlighted here.

Max page length for abstract 1.5 pages (no more than 600 words)

Please write about the capabilities of the individual components in the respective sections later in the report.

Part 1 – Technical report

Think about this part as writing a detailed developer guide. However, unlike CS2103, this part is written more like report. Clarity of thought and logical organization of ideas are important rather than information dump.

# 1 SPA Design

Give an overview of your main SPA components and the way they interact. You can use architecture, component diagrams, and component-level sequence diagrams here to explain the overall structure if your SPA implementation. Do not repeat the information from course material (e.g., lecture notes, requirements etc.)

Describe each component separately in a subsection if its own. Don’t use more than three levels of nesting. Do not use C++ code snippets or pseudocode for explanation. Use correct class/object diagrams to show the structure and activity/sequence diagrams to show the workflow or interaction instead.

At the end of each subsection, document important design decisions. Follow guidelines in lecture notes and in-class activities to analyse and justify. Do not go on to explain each row/column in the table (if you choose to use a tabular comparison) in the description. The description should only justify why you chose the alternative despite its shortcomings, if any.

You should discuss design decisions related to:

1. Architectural changes
2. Parsing and validation
3. Data representations for design abstractions in PKB
4. [Iteration 2/3] Solutions to speed up information access in PKB
5. Query evaluation
6. [Iteration 2/3] Optimisation strategies
7. Any other issue that is important

Have one subsection which briefly introduces a sample SIMPLE program that you can use as a running example in the entire report.

In general, choose the top three items per component that you think are important and showcases your technical report writing skill to explain in these subsections.

**Page length limitation** – 45 pages in the final report (which means iteration 1 and 2 should be appropriately scaled, OR, in iteration 3 you need to trim earlier portions appropriately)

## Sample SIMPLE program

Briefly describe and provide a listing of a sample SIMPLE program here. You can use this simple program as a running example throughout your report. This means you will have to think of one SIMPLE program that suffices to explain all things your SPA implementation provides.

## 1.2 Source processor

Describe how you parse and validate SIMPLE source code using example from Section 1.1 at a high level. Use sequence diagrams or activity diagrams as appropriate.

Describe when and how you extract (design) information and store the information into the PKB.

Also, describe the interaction between the source processor and PKB, possibly using an appropriate example from the program in Section 1.1, using sequence and/or activity diagrams. **Choose the interactions such that you can document integration tests for these interactions in Section 2.**

Describe the parsing, validation and extraction process you implemented here.

Things to note:

* Ensure the accuracy of the sequence/activity diagrams.
* Document the abstract API appropriately (Section 1.2.2)
* Refrain from using specific method calls from your implementation to explain the parser or the design extractor.

Indicate if you used any design patterns here and the benefits you obtain here.

### 1.2.1 Design decisions

Compare the design considerations of your implementation described above and justify your choice in the implementation. Be succinct. You can use the tabular technique discussed in the lectures to compare; description should mainly contain the justification.

### 1.2.2 Abstract API

Document the most important API provided by the source processor. (You will provide a complete list of abstract API in the appendix).

Include the API used for interacting with the PKB

|  |  |
| --- | --- |
| API | 1-2 line description |
| … | … |

## PKB

Describe your PKB structure for storing entities, relationships and any other auxiliary information. Please don’t repeat information here (e.g., explaining how you store one relation suffices; no need to explain all relations you store). It suffices if you can draw schematic diagrams; consider using examples from the sample SIMPLE program to explain the structure.

When you describe the PKB interaction with source processor or query processor, choose the interactions such that you can document integration tests for these interactions in Section 2.

Indicate if you used any design patterns here and the benefits you obtain here.

### 1.3.1 Design decisions

Again, compare the different design considerations for PKB data structure(s) and data retrieval. Use a similar format as Section 1.2.2

### 1.3.2 Abstract API

Document the most important API provided by the PKB; categorize them as API for source processor interaction and API for query processor interaction. (You will provide a complete list of abstract API in the appendix). If you have explained the relevant API when explaining other components, please don’t repeat them here.

## 1.4 Query processor

The query processor contains the Pre-processor (Parser + Validator) and Query Evaluator. Describe how you parse and validate PQL queries. You should focus on how types are validated here. Do not provide procedural description (pseudo-code) on how validation is done; you may choose to use activity diagrams to do so.

Describe the data representation for program queries.

Describe your query evaluation strategies and optimizations done. Show up to three examples of how evaluation is done and its intermediate/final results table.

You can choose to have the pre-processor and query evaluator in two separate subsections.

Describe the interaction between the query processor and PKB using examples from Section 1.1 to show how different types of clauses are evaluated. Use sequence and/or activity diagrams as appropriate. **Choose the interactions such that you can document integration tests for these interactions in Section 2**.

Indicate if you used any design patterns here and the benefits you obtain here.

### 1.4.1 Design decisions

Compare the different considerations for query evaluation and optimization. Justify your choice.

### 1.4.2 Abstract API

Document the most important API provided by the query processor; categorize them appropriately. (You will provide a complete list of abstract API in the appendix).

# 2 Testing

Describe the technical aspects of testing in this section.

When you want to use example test cases, focus on the valid tests and document the following for the test case.

1. Test Purpose: Explain what you intend to test in this test case
2. Required Test Input(s): Explain what component (or the whole system) you test, and what input must be fed to this test case
3. Expected Test Result(s): Specify the results to be produced when you run this test case
4. Any other requirements for running this test case (e.g. to run a test case for a component, you may need to implement a simulated environment)

**Page length limitation** – 25 pages in the final report (which means iteration 1 and 2 should be appropriately scaled, OR, in iteration 3 you need to trim earlier portions appropriately).

## 2.1 Unit testing

Provide two sample unit test cases for PKB and two sample unit test cases for query processor. Select test cases such that different parts of the PKB/query processor are tested.

If you wish to include additional samples, please place them in the appendix (graders are not obliged to read the appendix).

## 2.2 Integration testing

Describe how you tested the integration of each major component in SPA. Provide an example of **integration test cases that test the example interactions listed in Section 1**.

If you wish to include additional sample, please place them in the appendix (graders are not obliged to read the appendix).

## 2.3 System testing

Explain how you have designed your system test cases (think about the in-lecture activities in Week5, particularlyTESTING-2 to TESTING-4). Pay equal attention to SIMPLE source code and PQL queries used for system test cases. Use examples from Section 1.1.

Provide two sample test cases in AutoTester format.

If you wish to include additional sample, please place them in the appendix (graders are not obliged to read the appendix).

## 2.4 Any other testing [optional]

Give examples of stress and scalability testing, if any.

# 3 Extensions to SPA

## [Iteration 2]

Describe your proposed extension. It should contain:

1. Definition of the extension
2. Overview of the possible implementation challenges
3. How they affect the SPA components

## [Iteration 3]

Explain the extension. It should contain:

1. Implementation changes to SPA component/design/interaction design
2. Abstract API
3. System testing with one or two example test cases

For the final report, you may remove the “Overview of the possible implementation challenges” of Iteration2 and include the actual challenges. Treat this section as a mini-report for the extension.

**Page length limitation** – 10 pages in the final report (which means iteration 2 should be appropriately scaled).

Part 2 – Project management

# 4 Planning

Describe the task allocation to each team member in a succinct tabular form. Use the following format.

|  |  |  |
| --- | --- | --- |
| Component | Team Member  (Additional responsibilities, if any) | Allocation |
| Source processor | ABC  (Team lead) | * Implementation of lexer * Validation of call statements * Parser for container statements |

Identify all the tasks involved in the project using a succinct table. An example is provided below.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Task id | Task | | Activity id | | | Activity | | | |
| 1 | Implement SIMPLE parser | | 1.1[P] | | | Research Recursive Descent Parser & Regex Parser | | | |
| 1.2[C] | | | Implement tokenisation | | | |
| 1.3[P] | | | Plan Front-end API | | | |
| 1.4[D] | | | Document front-end abstract API | | | |
| 1.5[T] | | | Unit test tokenization | | | |
| 2 | Create PKB data structure | | 2.1[P] | | | Research on the data structure for Next/Next\* | | | |
| 2.2[C] | | | Create data structure for Next/Next\* | | | |
| 2.3[C] | | | Implement insertion and retrieval API for Next/Next\* | | | |
| 2.4[T] | | | Unit test insertion API for Next | | | |
| 2.5[T] | | | Unit test retrieval API for Next\* | | | |
| Color code | | | | Source processor | | | | PKB | |
| Legend: | | [D] Documentation activity | | | [P] Planning/design activity | | [C] Coding/ Implementation activity | | [T] Testing activity |

You are free to choose any reasonable representation. However, ensure that your representation eases your work to generate and represent the project plan.

**Page length limit** – 12 pages.

## 4.1 Project plan

Use Gantt chart or other visual aid to show the task coverage across the iterations. Show the major activities done in the iteration for the task selected on a per-week basis. Include testing activities too in the project plan.

You can use tools like the following to generate the Gantt chart.

* <https://templates.office.com/en-sg/simple-gantt-chart-tm16400962>
* <https://www.teamgantt.com/>

It may help you to color-code the tasks and then use the same colors in the gantt chart to show correspondence. A crude example is shown below.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Iteration 1 | | | | | | | | | | |
| Team member | Activities | | | | | | | | | |
|  | Week2 | | Week3 | | Week4 | | Week5 | | Week6 | |
| ABC | 1.1 | 1.1 | 2.1 | 2.2 | 2.4 |  |  |  |  |  |
| XYZ | 1.2 | 1.5 | 1.5 |  |  |  |  |  |  |  |
| PQR | 2.1 | 2.1 | 2.3 | 2.4 | 2.5 |  |  |  |  |  |

You can generate more beautiful gantt charts using any of the available tools.

# 5 Test strategy

Describe your test strategy briefly. You can include some of the following.

* How do you use automation in testing?
* How do you track the defects through its lifecycle once it is reported?
* What is the average defect resolution time?

Provide a test plan that explains the testing activities you have conducted during the iteration(s). In particular, include when you planned for integration and system tests. You can use a format similar to the project plan above; focus only on the testing activities. Ensure that your test plan is in sync with the project plan.

**Page length limit** – 8 pages

# 6 Coding standards

State the coding standard you followed and document any deviations from the adopted standard. Show one short example of following the standard you adopted. This section should be as brief as possible.

# 7 Correspondence of the abstract API with the relevant C++ classes

Consider these questions:

1. Do you use same naming conventions in abstract API and your C++ program?
2. Do you use typedef to map abstract API type names to C++ types?

Give up to 3 examples.

**Page length limit** (Section 6 and 7 combined) – 3 to 5 pages

Part 3 – Conclusion

# 8 Reflection

Don’t make this section a laundry list of all things you did in the course of the semester.

When you write this section, consider the following questions. This is not a ranting section. Be professional in your writing here. State facts.

Discuss problems encountered that affect project schedule. In which areas do you plan to improve in the next iteration or project?

What worked fine for you? What was a problem?

What management lessons have you learned?

Discuss any other project experiences and issues.

**Page length limit** – 2 pages

Appendix

This section is not directly graded. The teaching team may refer only if there is a need.

No limit on the page length here. However, be reasonable!

# 9 PKB Abstract API

[Compulsory Appendix]

Write APIs for the design abstractions listed below.

1. VarTable
2. ProcTable
3. AST (if any)
4. Follows, Follows\*
5. Parent, Parent\*
6. Modifies for assignment statements
7. Uses for assignment statements
8. [Iteration 2/3] Calls, Calls\*
9. [Iteration 2/3] Next
10. [Iteration 2/3] Next\*, Affects, Affects\*

Note that APIs for Follows/\*, Parent/\* can be defined separately from AST or as part of APIs involving AST.

Please follow the examples from course materials to document your API.

Remember to use abstract types and describe normal and abnormal behaviours.

# 10 API Discovery Process

[Good-to-have Appendix]

To help you discover your APIs for PKB and Query Evaluator, start by answering the numbered questions. You can answer ONE question in this appendix to show the thought process in coming up with the APIs. You do not need to answer more than one question in this appendix, but they can still be used in discovering your APIs.

Using subsections, describe how SPA components work with design abstractions (possibly via APIs), following examples in course materials using English.

Feel free to shorten/abbreviate repeated steps. For each interaction, continue description until you feel you are ready to switch to document a given API. Include as many steps as you think it is enough for you to get useful feedback.

Base your answers on the following SIMPLE source program and queries.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| procedure main {  1. read x;  2. read y;  3. while (y != 0) {  4. x = x / y;  5. read y; }  6. print x; } | assign a; stmt s; variable v;   |  |  | | --- | --- | | 1 | Select a pattern a(“x”, \_”y”\_) | | 2 | Select s such that Follows (1, s) | | 3 | Select s such that Follows (s, 3) | | 4 | Select s such that Follows\* (1, s) | | 5 | Select s such that Follows\* (s, 3) | | 6 | Select s such that Parent (3, s) | | 7 | Select s such that Parent (s, 5) | | 8 | Select s such that Parent\* (3, s) | | 9 | Select s such that Parent\* (s, 5) | | 10 | Select v such that Modifies (1, v) | | 11 | Select a such that Modifies (a, “x”) | |

Questions:

1. Write a sequence of steps describing how Parser works with ProcTable & VarTable.  
   For example,
   1. Insert “main” to ProcTable, return index
   2. Insert “x” to VarTable, return index
2. Write a sequence of steps describing how Parser builds an AST when parsing procedure Main.
3. Describe how Parser works with Follows and Parent

For example,

* 1. Parser must set Follows (1, 2)

1. Describe how Query Processor works with AST and VarTable when evaluating Query #1  
   Hint: Query Processor must traverse the AST in depth-first order and find subtrees that match pattern.

For example, you could start off as follows:

* 1. Get root of procedure main, return p
  2. Get first child of p, return p
  3. isMatch(p,”assign”) – no
  4. etc.

1. Describe how Query Processor works with Follows/\* when evaluating Query #2, #3, #4, and #5.
2. Describe how Query Processor works with Parent/\* when evaluating Query #6, #7, #8, and #9.
3. Describe how Program Parser works with Modifies when parsing statements 1 and 4.
4. Describe how Query Processor works with Modifies when evaluating Query #10, and #11.