2Software Engineering Stage 6 (Year 12) – Sample Software Engineering Systems Report Template

Tempe High School

Contents

1. Identifying and defining 2
    1.1. Define and analyse problem requirements 2
    1.2. Tools to develop ideas and generate solutions 3
    1.3. Implementation method 4
    1.4. Financial feasibility 5
2. Research and planning 7
    2.1. Project management 7
    2.2. Quality assurance 8
    2.3. Systems modelling 10
3. Producing and implementing 21
4. Testing and evaluating 22
    4.1. Evaluation of code 22
    4.2. Evaluation of solution 23

1. Identifying and defining

1.1. Define and analyse problem requirements

Problem context

*Students **analyse** the problem by **describing** each of its individual components and **explaining** how each of these components contribute to the problem needing resolution.*

Currently, students use a password manager that is not secure or is built-in a browser. This problem will be a stand-alone solution that will NOT be built-in to avoid some of the vulnerabilities associated with browsers.

Needs and opportunities

Students **describe** the needs of the new system to be built based on the problem context and using the table given below.

Need	Description
1. Storing passwords safely	Storing passwords in an encrypted format so that password cannot be read in a plain text format.
2. Generating passwords	Where the user signs up to a new website, the app will generate a secure password of at least 12 characters.
3. Autofill	Be able auto-fill passwords manually after user clicks a prompt (on verified websites only)

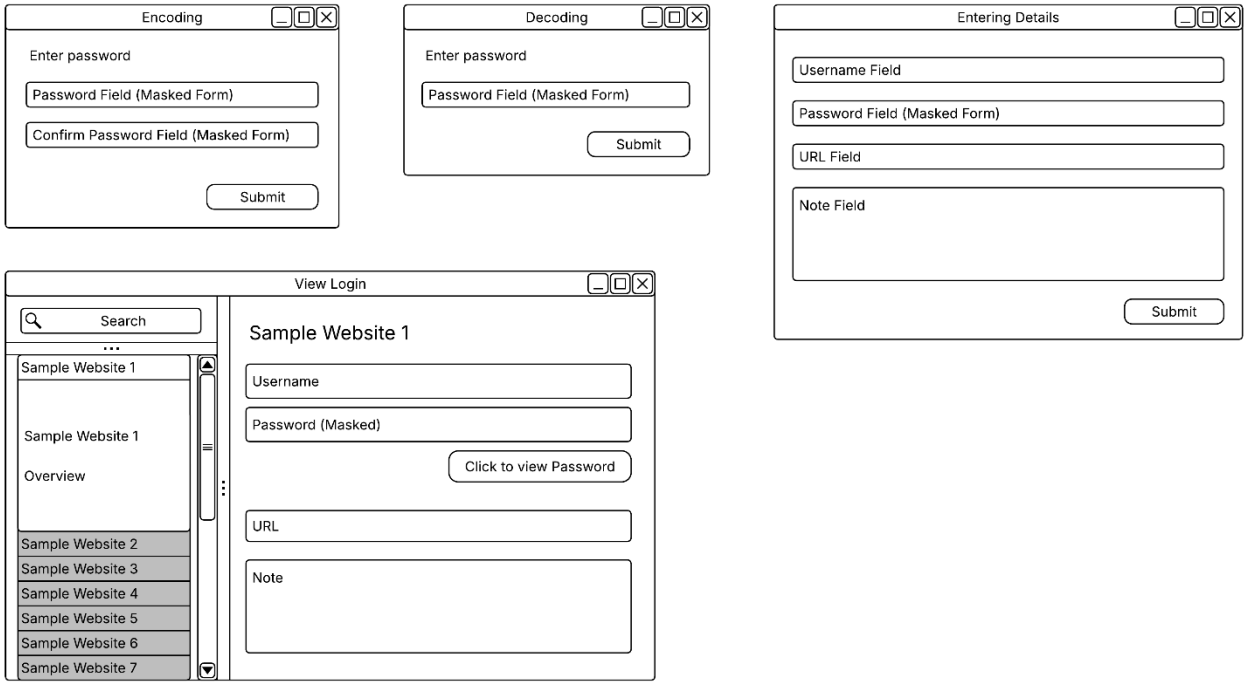
Boundaries

Students **analyse** any limitations or boundaries in which this new system will need to operate. Boundaries can include but are not limited to: hardware, operating systems, security concerns etc.

1.2. Tools to develop ideas and generate solutions

Application of appropriate software development tools

Students **apply** appropriate tools such as brainstorms, mind-maps, storyboards and prototypes.



1.3. Implementation method

Students explain the applicability of the implementation method for the current project. These methods are normally direct, phased, parallel, or pilot.

Implementation method: Pilot

My justification for doing the pilot implementation method is that only a select number of users that are willing to ‘beta test’ the new version of the application. If the new update of the application works without breakages or immediate bugs, then the new update will then be pushed to all users.

If there is a security vulnerability that is discovered when the application is in deployment, then a direct implementation method will then be used, as to maintain security and not to potentially expose any passwords as well as sensitive data.

1.4. Financial feasibility

Students are to **conduct** a financial feasibility study, including producing an opening-day balance sheet, to assess whether their application is financially viable.

SWOT Analysis

Strengths	Weaknesses
• Password managers is not hosted on third-party servers where they can potentially get breached.	• Locally storing passwords in inherently risky as it can be accidently deleted, misplaced, or stolen.
Opportunities	**Threats**
• Some businesses who are not that tech-focused in their operations may not store their passwords in a secure format or each employee may use the same password for each of the web services they use.	• IT-focused businesses are more than likely to hire a IT manager to make and maintain their own in-house tech, reducing the need for them to outsource it to me.

Study	Go or No Go?	Assessment and evidence
Market feasibility	No Go	There are already well established open-source and closed-source password managers out there (e.g. KeePassXC).
Cost of development	Go	The initial phase of development may be difficult as I must design the program, code it, run it on my machine, and test it on many computers and operating systems. This takes a lot of time. If the client wants the

Study	Go or No Go?	Assessment and evidence
		program within a brief time limit, then an initial payment may be needed depending on their requirements.
Cost of ownership	Go	If the clients want to host the passwords on an offsite server and not locally in the computers, then servers may be hired by the clients online (e.g. Amazon Web Services) or host their own servers. Since that I am the one who owns the software, I am the only who can troubleshoot and diagnose issues. This adds costs once maintenance starts
Income potential	No Go	Clients that are businesses can just simply hire an IT person to manage their own password managing software own their own servers, potentially making it harder for businesses to hire me.
Future expansion opportunities	Go	If I learn new skills that can help new clients, I can offer them new services as a combination of software.

Opening Day Balance Sheet

2. Research and planning

2.1. Project management

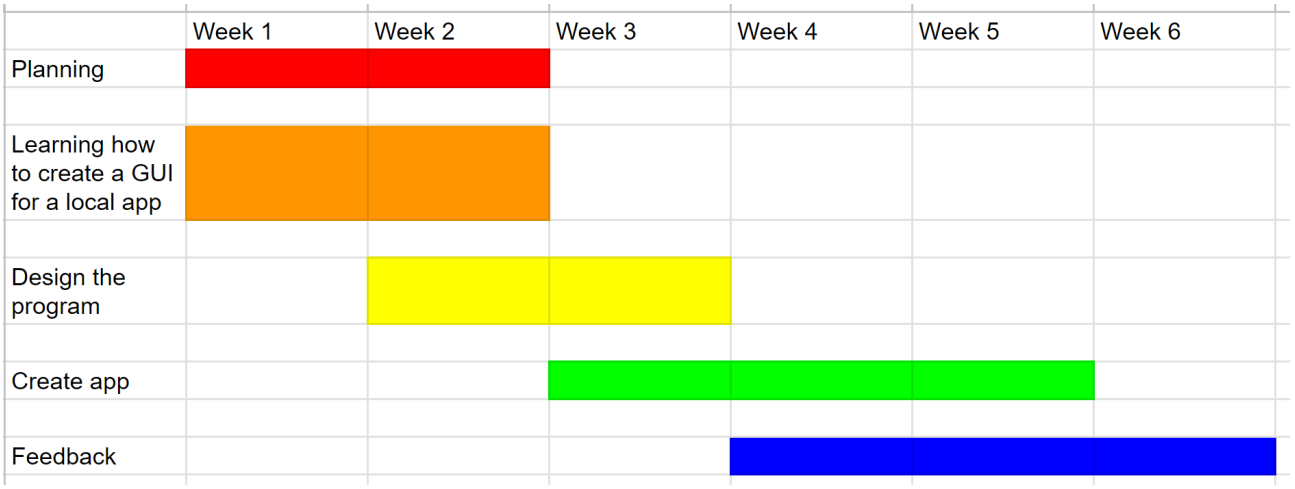
Software development approach

Students **explain** the software development approach most applicable for this current project. These are normally: Waterfall, Agile and WAgile.

Agile: it allows flexibility and adaptability that doesn't happen with other development approaches. It also allows clients to add functionality and features that they might want during development.

Scheduling and task allocation

Students **develop** a Gantt Chart that details the tasks required to be completed, person or people assigned to each task, timeline that does not exceed the project due date, resources required. In addition, students **identify** any collaborative tools used. For example Repl.it, GitHub and so on.



2.2. Quality assurance

Quality criteria

Students **explain** quality criteria based upon the needs from Section 1.1. These quality criteria should contain qualities, characteristics or components that need to be included or visible – based on Section 1.1. – by the end of the current project.

Quality criteria	Explanation
1. Storing passwords safely	Be able to store the passwords in a encrypted format, and not have the password be able to be read in plaintext.
2. Generating passwords	Generating secure passwords with a minimum amount of characters in order to maintain a minimum level of entropy.
3. Autofill	Have the application auto-detect the browser has a login field if one of the URL in the login details is present.

Compliance and legislative requirements

Students **explain** compliance and legislative requirements their projects need to meet and how they plan to mitigate them where possible. For example, projects that deal with sensitive personal data being publicly available may fall under the Australian NSW Privacy and Personal Information Act (1998) and/or Federal Privacy Act (1988). Alternatively, international standards on information security management such as ISO/IEC 27001 may also be applicable.

Compliance or legislative issue	Methods for mitigation
NSW Privacy and Personal Information Act (1998)	

Compliance or legislative issue	Methods for mitigation

2.3. Systems modelling

Students are to **develop** the given tables and diagrams. Students should consult the Software Engineering Course Specifications guide should they require further detail, exemplars or information. Each subsection below should be completed with Section 1.1. in mind.

Data dictionaries and data types

Students take the needs identified in Section 1.1. of this Systems Report. For each need, students **identify** the variables required, data types, format for display, and so on.

Need
1. Storing passwords safely

Variable	**Data type**	**Format for display**	**Size in bytes**	**Size for display**	**Description**	**Example**	**Validation**
Pasword	String	XX…XX	64	0	Password for encryption	P@ssw0rd123	

Need							

Need							
2. Generating passwords							
Variable	**Data type**	**Format for display**	**Size in bytes**	**Size for display**	**Description**	**Example**	**Validation**
Password length	integer	XX	2	2	What will the password length be for the new password	16	<10

Need							

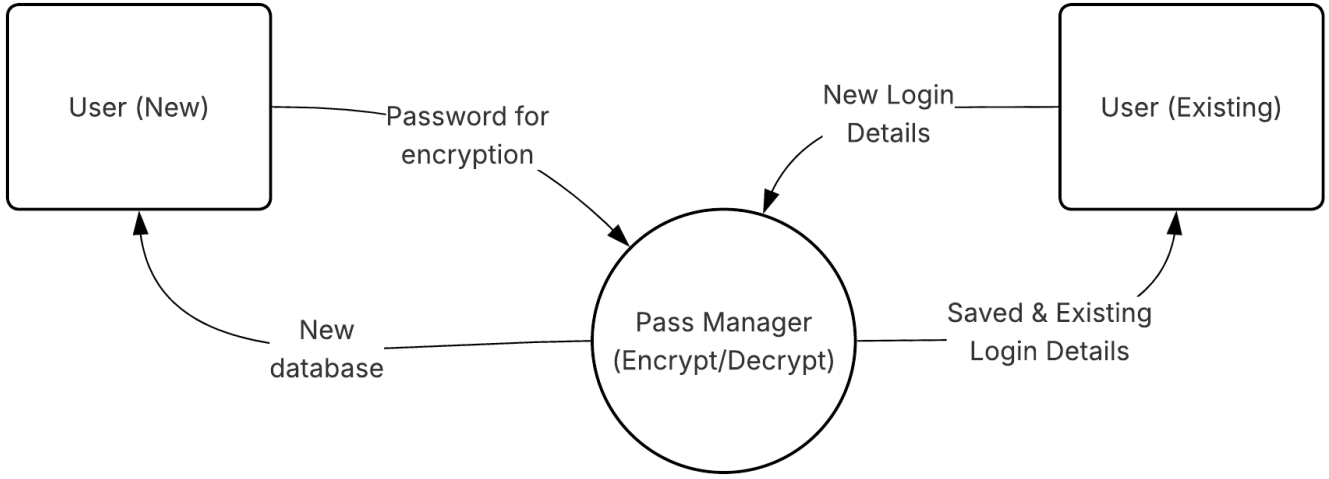
Need							
3. Autofill							
Variable	**Data type**	**Format for display**	**Size in bytes**	**Size for display**	**Description**	**Example**	**Validation**
URL	string	https://XXXX.XXX/	256	0	URL to detect for the log in details	http://mail.google.com/	Valide URL structure

Need							

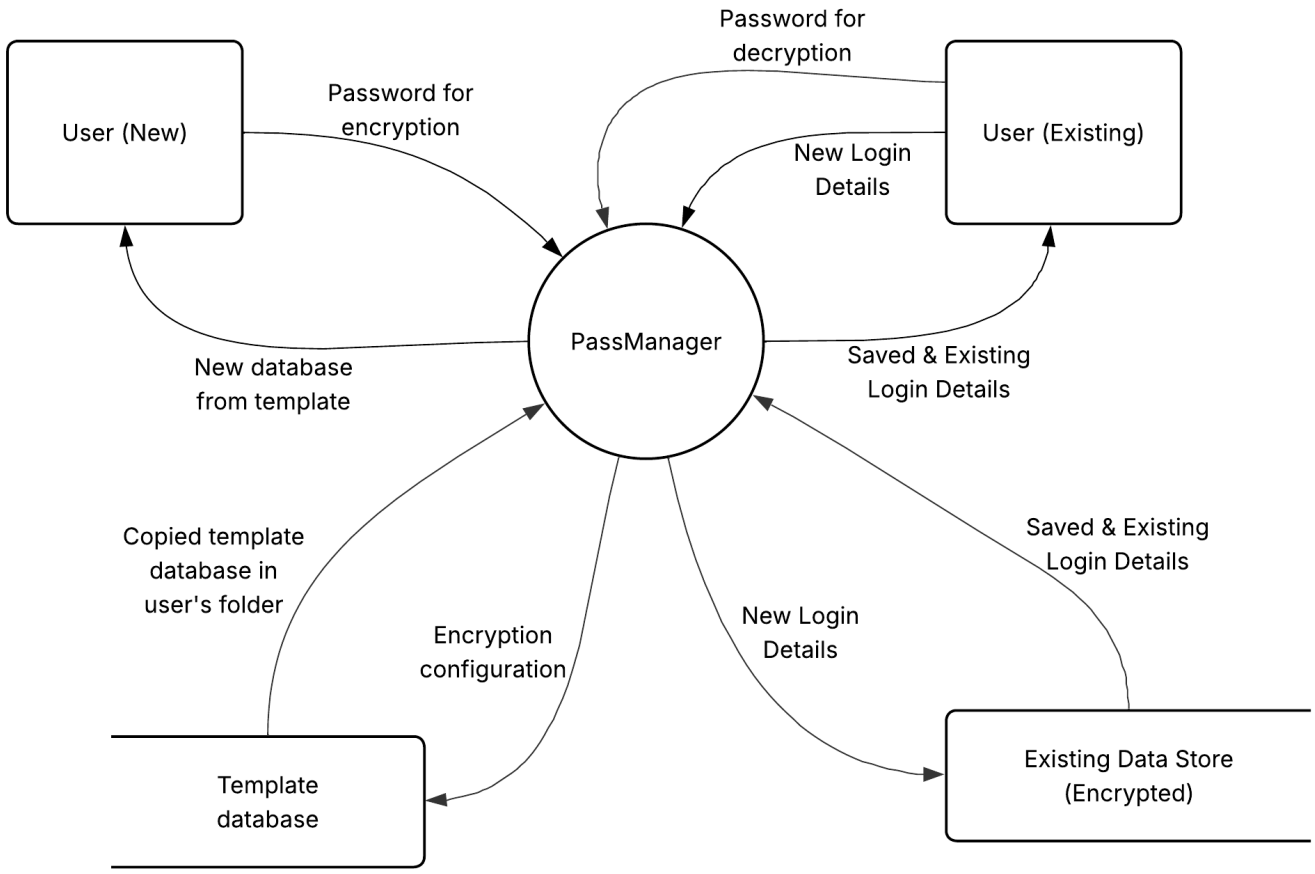
Data flow diagrams

Students **develop** data flow diagrams (DFDs) at Level 0 and Level 1. These diagrams should explicitly include the variables from the data dictionaries previously identified as well as the needs identified in Section 1.1.

Level 0



Level 1

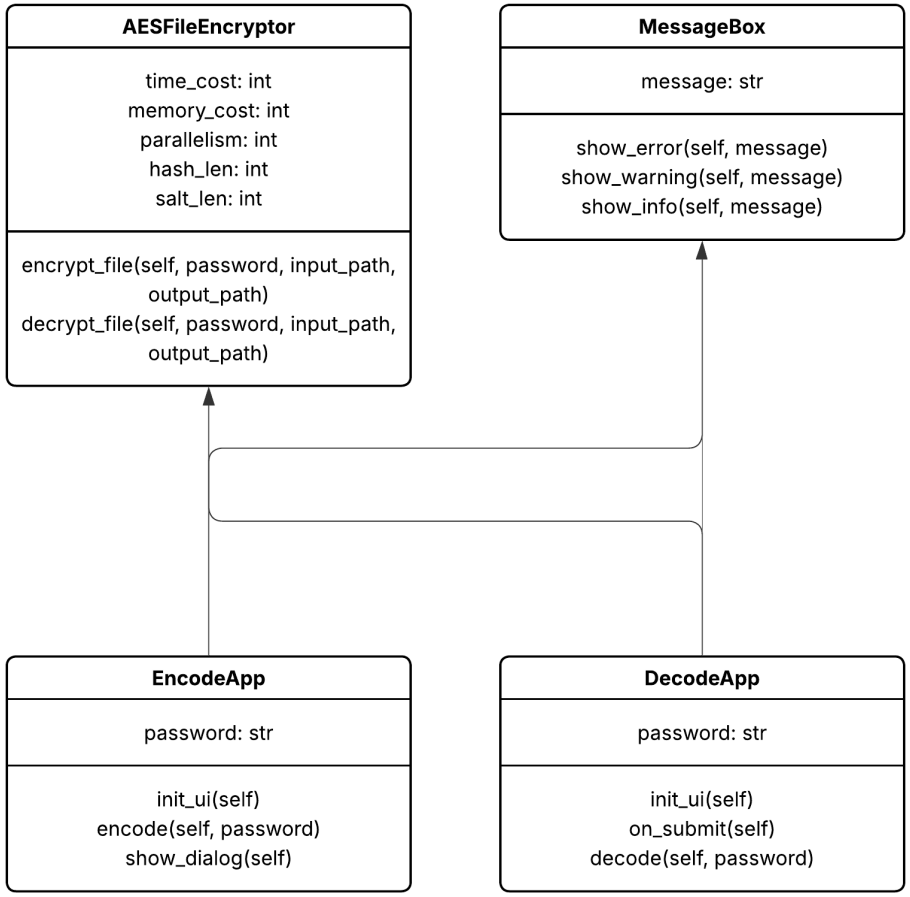


Structure charts

Students **develop** structure charts demonstrating how the procedures, modules or components of the final solution are interconnected.

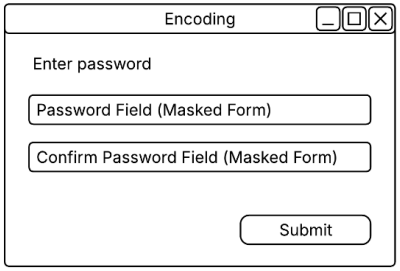
Class diagrams

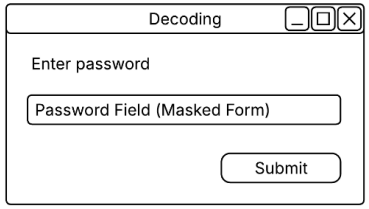
Students **develop** class diagrams demonstrating how each class is related to the other.

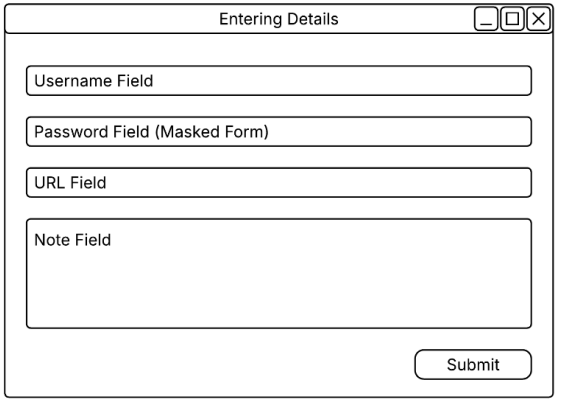


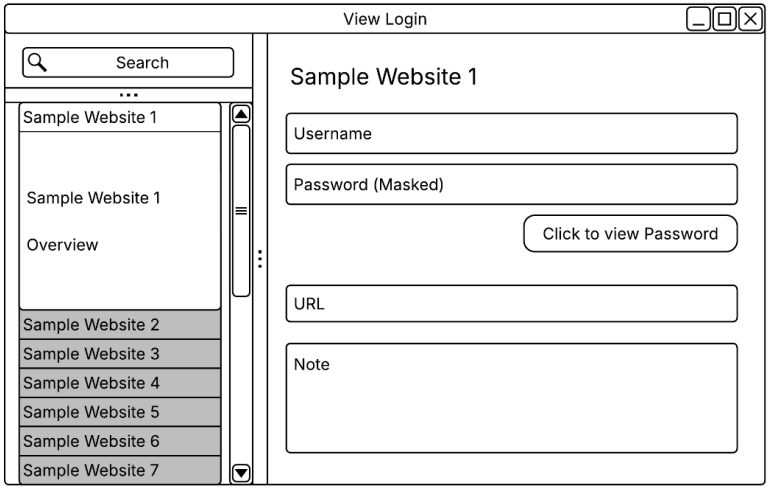
Storyboards

Students **develop** storyboards, visually representing the software solutions they will build.









Decision trees

Students **develop** decision trees to visually outline the logic flow and chain of decisions or selections the final solution will need.

Algorithm design

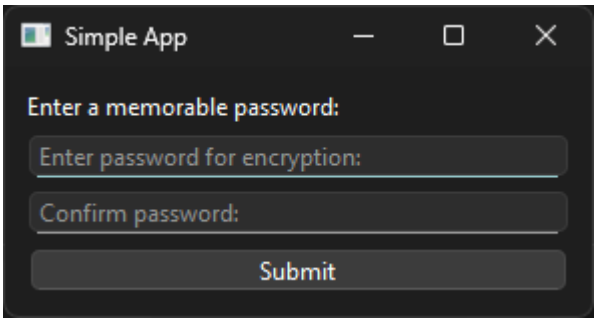
Students **develop** algorithms using methods such as pseudocode or flowcharts to solve the problem and meet the needs from Section 1.1. These algorithms should explicitly include the variables from the data dictionaries created in the previous section.

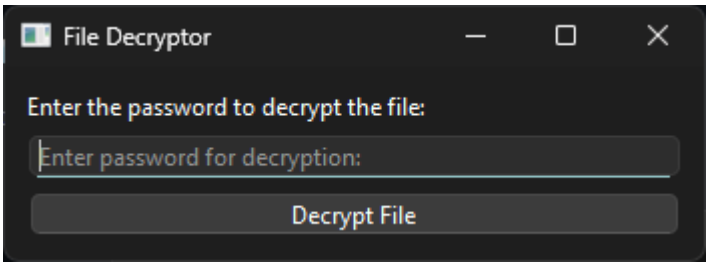
3. Producing and implementing

Solution to software problem

Students are to **include** screen shots of their final developed solution here. Each screenshot should include a caption that **explains** how it links to the:

- Needs identified in Section 1.1.
- Components of Section 2.3. such as the storyboards, data dictionaries and so on.





Version control

Students **describe** what version control system or protocol was implemented.

No version control was implemented.

4. Testing and evaluating

4.1. Evaluation of code

Methodology to test and evaluate code

Students **explain** the methodologies used to test and evaluate code. Methodologies include:

- Unit, subsystem and system testing
- Black, white and grey box testing
- Quality assurance.

I have used subsystem and system testing for this application, I used subsystem testing for each of the components of the application such as the key derivation function and the file encryption. I used system testing for the whole application, and did a flow on how user would use the application.

Code optimisation

Students **explain** the methodologies used to optimise code so that it runs faster and more efficiently. Methodologies include:

- Dead code elimination
- Code movement
- Strength reduction
- Common sub-expression elimination
- Compile time evaluation – constant folding and constant propagation
- Refactoring

Only refactoring wass used in order to reduce bottleneck in installing the program.

4.2. Evaluation of solution

Analysis of feedback

Students **analyse** feedback given to them on the new system they have just created. This feedback can be in the form of an interview, survey, focus group, observation or any other applicable method. Students should also include overall positive, negative or neutral sentiments towards the new system in their response.

Neutral responses; users did not test the full application of what was meant to be delivered.

Testing methods

Students **identify** the method or methods of testing used in this current project. For each they use, students are to **explain** how and why it was used.

Method	Applicability	Reasoning
Functional testing	Yes	Each of the function were need.
User Acceptance testing	No	
Live data	Yes	Sample passwords were given in testing.
Simulated data	No	The installation environment will never be controlled.
Beta testing	No	Beta-testing is should be used in a secure application in case of major vulnerabilities.
Volume testing	No	This application will only be run on a computer locally.

Security Assessment

Students are to **perform** an extensive security assessment of their final application and e**xplain** the countermeasures implemented.

Threat	Countermeasure

Test data tables

Students **identify** variables which were used for either path and/or boundary testing. Students **develop** these test data tables based on their algorithms versus their real code. Students then **state** the reason for including said variables.

Boundary testing

Variable	Maximum	Minimum	Default Value	Expected Output	Actual Output	Reason for Inclusion
Password	N/A	N/A	None	Successful encryption	Successful encryption	

Analysis of solution against quality success criteria

Students are to take each quality success criteria from Section 2.2 and place it here. For each quality criteria, **analyse** the components of the solution that met or did not meet each quality criteria. Give reasons why each success criteria were or were not met.

Quality criteria	Met?	Analysis