|  |  |
| --- | --- |
|  | **Qatar University**  **College of Engineering**  **Department of Computer Science and Engineering** |

Senior Project Report

**<ToDo: Insert Project Title />**

**Project Group Members:**

<ToDo: StudentName1 (StudentId1)

StudentName2 (StudentId2)

StudentName3 (StudentId3) />

**Supervisor**: <**ToDo:** Supervisor Name />

**Co-Supervisor**: <Remove if not applicable/>

**20??**

This project report is submitted to the Department of Computer Science and Engineering of Qatar University in partial fulfillment of the requirements of the Senior Project course.

# Declaration

This report has not been submitted for any other degree at this or any other University. It is solely our work except where cited in the text or the Acknowledgements page. It describes work carried out by us for the senior project. We are aware of the university policy on plagiarism and the associated penalties and we declare that this report is the product of our own work.

< **ToDo:** For each student add the signature and the date of signing the declaration />

Student: Date:

Signature:

Student: Date:

Signature:

Student: Date:

Signature:

# Abstract

< **ToDo:**

* The abstract is a brief overview of your project and its objectives.
* It should present an accurate summary of the problem your project has addressed and a summary of your solution.
* Highlight key achievements and most important conclusions.
* The length of your abstract should not exceed 500 words.

/>

<**ToDo:** **Writing the final report**

* To produce the final report, you can follow the following recommended steps:
* Where appropriate, copy material from your interim report into the final report. Go through that material, and update it based on changes that have occurred in your project between last semester and now.
* Revise the Abstract and enhance it by adding the project’s key achievements and most important conclusions. The last paragraph should highlight the novelty of your design (e.g., what makes your design unique and what are the impacts of your engineered solution, etc.).
* Fill in all of the appropriate material required for the final report.
* Update the Table of Content, the List of Figures, and the List of Tables.
* Review the whole document to make sure that it is coherent and to ensure that it addresses all the requirements listed in the Project Guide and the Project Grading Rubrics. Also make sure that the tense used is the present tense and the past and not the future (e.g., avoid ‘we will’ or ‘system should’ and report what has been done) except in the Future Work section.
* Seek your supervisor’s feedback and address any issues raised.
* Note that the template is only provided as a guide. In consultation with your supervisor, you can add other sections to align it with the nature of your project.
* If you are using Word ‘Track Changes’ you must accept all the changes before submitting your report.
* Keep the ToDo instructions and only remove them from the **final revised report** that you will submit **after** the Senior Project presentation and after addressing the examiners feedback.

/>

# Acknowledgment

< **ToDo:**

* Acknowledge any assistance you received for your project.

/>

Table of Contents

[Declaration ii](#_Toc127951596)

[Abstract iii](#_Toc127951597)

[Acknowledgment iv](#_Toc127951598)

[List of Figures vii](#_Toc127951599)

[List of tables viii](#_Toc127951600)

[1. Introduction and Motivation 9](#_Toc127951601)

[1.1. Problem statement 9](#_Toc127951602)

[1.2. Project objectives 9](#_Toc127951603)

[1.3. Expected benefits and impacts on various contexts 9](#_Toc127951604)

[1.4. Market Research and Business Viability 10](#_Toc127951605)

[2. Background and related work 10](#_Toc127951606)

[2.1. Background 10](#_Toc127951607)

[2.2. Related work 11](#_Toc127951608)

[3. Requirements analysis 12](#_Toc127951609)

[3.1. Software development process 12](#_Toc127951610)

[3.1.1. Applying the software development process 12](#_Toc127951611)

[3.2. Functional requirements 12](#_Toc127951612)

[3.3. Non-functional requirements 13](#_Toc127951613)

[3.4. Assumptions 14](#_Toc127951614)

[3.5. Ethics 14](#_Toc127951615)

[4. Project Plan 15](#_Toc127951616)

[4.1. Project milestones 15](#_Toc127951617)

[4.2. Project timeline 15](#_Toc127951618)

[4.3. Anticipated risks 15](#_Toc127951619)

[5. Solution Design 16](#_Toc127951620)

[5.1. High-level design 16](#_Toc127951621)

[5.1.1. Alternative solutions and tradeoffs 16](#_Toc127951622)

[5.1.2. Selected solution overview 16](#_Toc127951623)

[5.1.3. High level architecture 17](#_Toc127951624)

[5.2. Structural model 17](#_Toc127951625)

[5.3. Behavioral model 17](#_Toc127951626)

[5.4. Database design 18](#_Toc127951627)

[5.5. User interface design 18](#_Toc127951628)

[5.6. Design patterns 18](#_Toc127951629)

[6. Implementation 19](#_Toc127951630)

[6.1. Hardware/software used 19](#_Toc127951631)

[6.2. Challenging issues and solutions 19](#_Toc127951632)

[7. Testing and evaluation 20](#_Toc127951633)

[7.1. Functional testing 20](#_Toc127951634)

[7.2. Non-functional testing 20](#_Toc127951635)

[8. Conclusion 21](#_Toc127951636)

[9. Future work 22](#_Toc127951637)

[10. Student reflections 23](#_Toc127951638)

[References 24](#_Toc127951639)

[Appendix A – Use cases specification 25](#_Toc127951640)

[Appendix B – Test cases specification 26](#_Toc127951641)

[Other Appendices 27](#_Toc127951642)

# List of Figures

[Figure 1. Use cases diagram 13](#_Toc127949606)

# List of tables

[Table 1. Expected benefits and impacts on various contexts 10](#_Toc127949795)

[Table 2. Use cases summary 13](#_Toc127949796)

[Table 3. Evaluation plan for the non-functional requirement 13](#_Toc127949797)

[Table 4. Project-related Software Engineering Code of Ethics and Professional Responsibilities 14](#_Toc127949798)

[Table 5. Milestone of the project 15](#_Toc127949799)

[Table 6. Risks 15](#_Toc127949800)

[Table 7. Hardware/software to be used 19](#_Toc127949801)

# Introduction and Motivation

The purpose of this introduction is to serve as the gateway to understanding the core problem that our project aims to tackle, showing its significance, the associated obstacles and a basic overview of our proposed solution and its anticipated impacts.

In today’s digital age, the significance of a password cannot be underestimated, as it is the key that grants users’ authority over their online accounts. Regrettably, a substantial number of individuals have fallen prey to data breaches due to the widespread reliance on weak, default, or repeatedly used passwords. This unfortunate consequence can be traced back to a simple rationale that predictable passwords and password reuse make it easier to remember and convenient for users who might be unaware of the security risks hidden behind that convenience. This vulnerability becomes especially apparent when a single breach in one service jeopardizes multiple other accounts. Astonishingly, these poor password practices have contributed to a staggering 81% of corporate data breaches, highlighting the magnitude of the problem at hand.

This is where our password manager application comes in, aimed with the sole purpose of addressing the pervasive issues associated with weak, recycled and default passwords. Our goal is to provide users with a seamless and efficient experience that encourages them to protect their privacy and secure their accounts across all the services they use. Our password manager aims to fill in the gap between security and user convenience.

By providing an intuitive interface, secure password storage, and the capacity to generate robust passwords, we aim to provide users with a tool that elevates their online security without sacrificing the ease and comfort they seek. With our application at their disposal, users can confidently navigate the landscape of the digital realm, secure in the knowledge that their accounts and sensitive data remain well-guarded.

## Problem statement

The problem at hand is to design and develop a secure and user-friendly password manager application that combines some of the most desired and unique features of other products currently in the market while addressing the pervasive issues of weak, reused, and default passwords. The goal is to empower users to enhance their online security while maintaining the convenience they desire by offering a seamless and efficient experience for managing their passwords across various online services.



Technical Challenges:

* Implementing Robust Encryption: Developing strong encryption mechanisms to securely store and transmit sensitive user passwords.
* User Authentication: Ensuring secure and user-friendly methods for user authentication
* Password Generation: Creating an effective password generation algorithm that produces strong, unique passwords.
* Usable User Interface: Designing an intuitive and user-friendly interface for managing passwords, suitable for all users.
* Password Backup and Recovery: Implementing a secure and user-friendly password backup and recovery methods.
* Security Auditing: using tools for regular security audits and alerts for password-related vulnerabilities.
* Compliance with Privacy Regulations: Ensuring the application complies with data protection regulations and user privacy requirements.

Non-Technical Challenges:

* User Education: Educating users about password best practices and the importance of using a password manager.
* User Trust: Building trust in the security and privacy of the application, as users entrust it with their sensitive data.
* User Experience: Balancing security with user experience to create an application that users find easy and pleasant to use.
* Adoption and Advertising: Encouraging users to adopt our password manager and effectively highlight its benefits.

**Problem Classification based on its Computing Complexity**

Conflicting Technical Issues Tradeoffs:

* + The application will need to balance usability with security. Creating a user-friendly interface while ensuring robust encryption and protection against breaches is a technical challenge.
  + Choice of handling user data either locally or by using cloud storage requires extensive thinking and measuring of trade-offs between security and accessibility across devices.
  + Decisions such as the choice of encryption algorithms, key management, and password storage methods involve trade-offs between security and performance.

No Obvious Solution:

* + Designing a password manager that provides both convenience and strong security is not straightforward.
  + The best approach to user authentication, password storage, and secure password sharing may require innovative thinking.

Including Many Component Parts or Sub-Problems:

* + A password manager involves various components, such as user interfaces, encryption modules, database management, and some kind of synchronization across devices.
  + Addressing each component's unique challenges and ensuring they work effectively together is essential.

Involving Multiple Disciplines:

* + Developing a password manager requires expertise in software development, cryptography, user experience design, and potentially will require legal and privacy considerations.
  + Cross-disciplinary collaboration is often necessary to create a robust optimal solution.

## Project objectives

Main Objective

Develop a User-Friendly Password Manager Application that aligns with the project’s goal of enhancing user convenience and security.

Sub-Objectives

* Design an easy-to-use UX that is suitable for a wide range of users and perform user testing to improve the interface based on feedback.
* Implement secure encryption for data storage that meets or exceeds industry best practices by utilizing well known encryption algorithms and libraries since data security is a fundamental aspect of the project's objectives.
* Develop a password generation functionality that generates strong, unique passwords by utilizing established password generation algorithms.
* Ensure that password data is consistently accessible on all devices the user desires to enhance user convenience and accessibility.
* Implement multi-factor authentication to enhance security and user confidence.
* Design alerts by utilizing automated security scanning tools and implementing alert mechanisms that inform the user of their password strengths and recommend possible solutions to create a stronger password.
* Evaluate the compliance with privacy regulations related to user data protection and how it is stored which will require continuous evaluation and monitoring throughout the project's lifecycle. This is essential for user trust and legal compliance.

## Expected benefits and impacts on various contexts

The importance of our password manager app project lies in its crucial role in addressing and mitigating several significant challenges related to secure password management, and data protection. This project tackles these issues head-on, offering concrete solutions that enhance security for individuals and organizations alike. According to the 2021 Data Breach Investigations Report, in 2019, 80% of hacking-related breaches were reportedly linked to passwords and stolen credentials [[1]](#Bookmark4). In addition, according to a late 2019 Harris Poll, 75% of Americans struggle to keep track of all their passwords [[2]](#Bookmark3). In today’s world enhancing data security by any means is crucial, our password manager will protect login credentials through encryption and secure storage, contributing to overall data security by preventing unauthorized access to accounts. A study by IBM security reported that the average cost of a data breach in 2020 was $3.86 million [[3]](#Bookmark2). Today, many cybercriminals rely on bad password management to successfully hack into an enterprise’s networks and systems. In 2020 research conducted by the Ponemon Institute, it was revealed that a significant 59% of organizations continue to depend solely on their employees' memory to manage their passwords [[4]](#Bookmark1). This situation highlights the challenge employees face when choosing passwords, as they tend to opt for ones that are easy to remember. Given the growing complexity of password strength requirements, enterprises are faced with the imperative task of equipping users with suitable tools, such as password managers. These tools not only store passwords securely but also facilitate password generation and automatic filling when necessary.

Personal Interest and Career Goals

We became interested in this project due to the increasing importance of cybersecurity in our digitally connected world, even though there are great password managers out there, we have decided to face the challenge of developing one ourselves and delve into security and software development while also aligning with our career goals. As the field of cybersecurity continues to evolve, we hope that our involvement with this project enhances our skills and knowledge as well as allowing me to contribute to the broader goal of making the digital world safer for everyone.

Table 1. Expected benefits and impacts on various contexts

|  |  |
| --- | --- |
| **Context** | **Expected benefits and impacts** |
| Individuals | Users can enjoy an enhanced digital security against cyber threats, reducing the risk of personal data breaches. A simplified and easy-to-use password management application will encourage users across the board to use its tools as a way of protecting their personal information while maintaining conveniency. |
| Organizations | Employees in all organizations can utilize our password manager to spend less time on password-related issues and more on establishing efficiency and effectiveness, thus increasing the organization’s productivity as a whole. In addition, our tool is valuable in the sense that it eliminates the need for employees to write down their passwords, which is a common insecure practice in most organizations. |
| Society | Usage of our password manager will contribute to a safer online environment, reducing the risk of identity theft and fraud as well as allowing users to adopt such responsible online practices. |
| Global | Our app can contribute to a thriving local cybersecurity industry, furthering Qatar’s technical regional capabilities and boosting competitiveness. |

## Market Research and Business Viability

< **ToDo:**

* Conduct market research to address the following:
* Describe the market need and the market size.
* Identify the target customers and their demographic.
* Describe the competing products and how does yours differ from that offered by competitors? Highlight the novel features of your product and the benefits it offers.
* Develop a business plan including your business model, pricing, marketing strategy to bring your product to market and make it competitive.

/>

# Background and related work

## Background

< **ToDo:**

* Briefly discuss the major concepts, issues, and key problems related to your project. This should give the reader the necessary background information to understand your project.
  + Provide a brief overview of the field or domain that your project is focused on. The key players, and any relevant history or trends.
  + Identify the major concepts, theories, or technologies that are relevant to your project. This could include a brief explanation of any terminology, methods, or frameworks that are necessary for understanding your project.

/>

## Related work

<**ToDo:**

* Discuss the related work and how other researchers or developers have dealt with the problem at hand. Cite appropriate references including relevant research papers. A list of 10 to 15 references should be adequate to include in this section. This can include academic papers, books, patents, or other relevant sources. Seek your supervisor assistance to find and select relevant related work.
  + Discuss the key findings and approaches of each reference and evaluate their strengths and limitations.
  + Identify any gaps or weaknesses in previous solutions and explain how your project aims to address these issues.
  + Discuss the novelty and contributions of your project, and how it builds upon or differs from previous work. This can include new methods, algorithms, or other novel aspects of your solution.
* At the end of this section, you should highlight how your project is different than previous approaches. It is highly recommended that you use a table to summarize the key similarities and differences between your project and the related work.

/>

# Requirements analysis

< **ToDo:**

* In this section, you are describing and justifying your selection of the most appropriate software development process for your specific project and presenting the project requirements.

/>

## Software development process

<**ToDo:**

* Discuss possible alternative software development processes that can be applied for your project.
* Select an appropriate software development process for your project.
* Provide sufficient justifications why the selected software development process is the most suitable for your specific project.
* Describe how you are intending to apply the selected software development process to your project.

/>

## Applying the software development process

</**ToDo:**

* Document and provide evidence on how you **applied** the adopted Software Development Process throughout the project to produce the solution. This should also be reflected in the project plan presented in section ‎4.
* Use in-text referencing to relevant sections of the report to provide evidence of the proper usage of the adopted software development process. For example, you can reference specific sections of the report that describe the requirements gathering process, design phase, testing process, or project management techniques.
* Discuss any related issues, challenges, difficulties, advantages or drawbacks, and lessons learnt during the software development process. For example, you can describe how the chosen methodology helped to address specific project challenges or how it fell short in some areas.
* This subsection is extremely important and required.

/>

## Functional requirements

< **ToDo:**

* Collect, document, and analyze the project requirements.
* The requirements specification should be done using suitable methods and tools such as use cases.
* You may use flowcharts to describe the sequence of actions and data flow of process(es) required for your project.
* Produce a use case diagram
* List all use cases in the use cases summary table.
* Produce detailed specifications for each use case and add it to Appendix A - Use cases specification.
* Once you identify and describe all the use cases, you need to prioritize them and decide your project scope by highlighting the use cases that you will design and implement.

/>

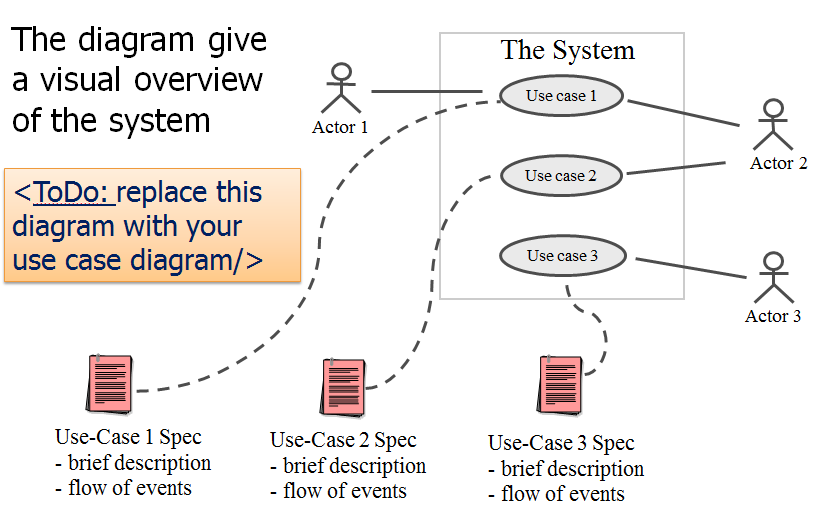


Figure 1. Use cases diagram

Table 2. Use cases summary

|  |  |  |
| --- | --- | --- |
| ID | Use case | Brief description |
|  |  |  |
|  |  |  |

## Non-functional requirements

<**ToDo:**

* Identify non-functional requirements (i.e., desired quality attributes) and clearly document them using measurable scenarios such that you can later test whether they are met.
* Categorize the requirements into different categories such as performance, reliability, security, usability, portability, and maintainability.
* Define measurable targets for each requirement. For example, if you have a performance requirement, you should specify the maximum response time or throughput. Instead of just saying the system should be fast, you should be more specific e.g., ‘Registering a new user should take less than 10 seconds, 95% of the time’. Instead of just stating that the system should be secure, you should specify what attacks must be prevented and what threats it must defend against.
* Ensure that your non-functional requirements are testable. In other words, make sure that you have a way to measure whether the system is actually meeting each requirement, such as through automated tests or user testing..
* Remember: If your non-functional requirement isn’t specific and measurable, then it isn’t a good non-functional requirement.
* Include an evaluation plan for each non-functional requirement, detailing how you will test whether the system is meeting the specified targets. This might include details on what tools or methods you will use to test each requirement, as well as any specific scenarios you will use to validate the system delivers the desired non-functional requirements.

/>

Table 3. Evaluation plan for the non-functional requirement

|  |  |  |
| --- | --- | --- |
| Non-functional requirement | Measurable targets | Evaluation Plan |
|  |  |  |
|  |  |  |

## Assumptions

<**ToDo:**

* List any specific project assumptions. Assumptions are potential failure points in a project. They need to be monitored and managed. A bulleted list could be used for this section.

/>

## Ethics

<**ToDo:**

* Ethical issues and professional responsibilities:
  + Identify the legal and ethical issues and professional responsibilities relevant to your project (e.g. proper handling of private information collected during the project, not to exaggerate the nature of work in the report rather the report should truly reflect the actual work completed, not to use someone’s work without citing proper reference, etc.). Consider the perspectives of diverse relevant stakeholders and articulate these with clarity and accuracy.
  + Discuss how you will address these identified ethical issues/concerns and responsibilities during the analysis, design, implementation, testing, evaluation, and delivery of your project.
* From [ACM and the IEEE Software Engineering Code of Ethics and Professional Practice](https://ethics.acm.org/code-of-ethics/software-engineering-code/) select the most relevant items to address the identified ethical issues/concerns and responsibilities in your specific project.
  + The selection should be based on appropriate project functionalities, the development tasks, the conduct of the team and interactions with other project stakeholders. Discuss and justify using/practicing them during the project work.
  + Consider alternative courses of action and justify your choices in logical manner with persuasive reasoning.
  + **Avoid** outlining general code of ethics and responsibilities not related to your specific project. **Do not cut and paste the entire ACM and IEEE codes of ethics into your report**. Instead, read through them, and in your report specifically address the ones most relevant to your specific project and discuss how you are considering them in your project.
  + At the end of your discussions, use a table to summarize the ethical issues/concerns and professional responsibilities relevant to your project.

/>

Table 4. Project-related Software Engineering Code of Ethics and Professional Responsibilities

|  |  |  |
| --- | --- | --- |
| Sec. No | Code | How to address the ethical issue / professional responsibility during your project |
|  |  |  |
|  |  |  |

# Project Plan

## Project milestones

< **ToDo:**

* In a table list and describe the major milestones of the proposed project plan. Milestones are checkpoints that have specific deliverables to produce.

For example, a milestone could involve choosing a use case (or a component from your high-level design) then designing, implementing, and testing its realization.

/>

Table 5. Milestone of the project

|  |  |  |
| --- | --- | --- |
| Milestone | Description | Deliverables |
|  |  |  |
|  |  |  |

## Project timeline

< **ToDo:**

* Project timeline defines who will do what and when. You need to:
* Breakdown each project milestone into manageable tasks. Using a table, provide a brief description of each task.
* Estimate the time required to complete each task.
* Allocate the tasks to individual team members and define a work schedule stating the estimated beginning and completion dates of each task. You can use Microsoft Project or Excel to create your project timeline. Insert a Gantt chart and/or a project timeline table in this section.
* Keep in mind the total number of days budgeted for the project. Also the role and the responsibilities of each team member should be clearly decided.
* Be sure to include time to evaluate your work, to reflect on the experience and to document your work at each milestone to incrementally produce your final report. You should include time to finalize and review the final report and presentation.

/>

## Anticipated risks

< **ToDo:**

* Present a table of risks highlighting the potential events that might result in failure to successfully complete the project.
* What is your approach to minimizing each risk?

/>

Table 6. Risks

|  |  |
| --- | --- |
| Risk event | Approach to minimizing the effect on project success |
|  |  |
|  |  |

# Solution Design

<**ToDo:**

* In this section you need to document the detailed design specifications of the software components to meet the functional and the non-functional requirements of your project.
  + You should document your detailed design using suitable methods and tools such as class diagrams, sequence diagrams, activity diagrams and state diagrams. Also, you need to provide detailed justification of design choices. Additionally, you should highlight the novel aspects of your design.
* During semester 1, you will design, implement, and test a Proof of Concept (PoC) to prove the feasibility of your design idea and deliver the core features of your solution by choosing the most important/critical use cases. It is expected that the PoC delivers 25% to 30% of the important use cases.
* Note that for every design aspects you will present in the sections below, you must:
* **Identify and evaluate possible alternative solutions** (i.e., design choices) and analyze their tradeoffs. You must provide sufficient justifications on why you have preferred and selected a particular solution or design choice over others.
* Justify your optimal design choice by **discussing how your design choices** enables you to achieve the desired system quality attributes documented in Section ‎3.3 Non-Functional Requirements.
* Further justify your design choice by discussing relevant **design principles** that influenced your system design. For example, highlight how your design comply with key software design principles such as:
* Separation of concerns: e.g., separation of the UI from the business logic.
* Abstraction: hide the component’s complexity behind simple interface.
* Modularity: divide the system into components.
* High cohesion: component’s functions should be functionally related.
* Low coupling: reduce dependencies between components.
* You could consider breaking the sub-sections listed below to clearly document and communicate your design.

/>

## High-level design

## Alternative solutions and tradeoffs

< **ToDo:**

* Identify possible alternative solutions to meet the project requirements and analyze their tradeoffs.

/>

## Selected solution overview

< **ToDo:**

* Present an overview of the selected solution. This can be done in the form of few paragraphs giving a birds-eye view of the solution.
* You must provide sufficient justifications on why your selected solution or design is better than alternative solutions.

/>

## High level architecture

< **ToDo:**

* Add a high-level architecture diagram of the proposed solution. The diagram should show how your solution is decomposed and organized into components. This should guide your detailed design.
* Describe the role and the interfaces of key components of your high-level architecture.
* Discuss and justify the architectural style (e.g., MVC, SPA) used by your solution.
* Discuss the key interactions between the identified components.

/>

## Structural model

<**ToDo:**

* **You may break this section into subsections** to document the detailed structural model of each major components or use case.
* The structure of your software components could be documented using:
* Class diagram for the whole system. If the model is too big partition the diagram using some reasonable criteria. For example, you may provide the entity classes, repository classes and the service classes as separate diagrams.
  + - * For every class, specify the declaration of all attributes (names, data types).
      * For every class, specify the signatures for all methods (i.e., parameter names and their types as well as the return type).
      * Show associations and aggregations between classes. You should define the association name and the multiplicities on both ends.
      * Show inheritance relationships between classes.
      * A brief explanation should accompany each diagram. Add few paragraphs discussing major design decisions such as how inheritance and polymorphism were used to improve the system design. Also briefly discuss how design principles such as the ‘Information Expert’ and ‘Whole controls the Parts’ principles were applied to your design.
* For the external services (i.e., boundary classes) the system interacts with (e.g., BankingService, MailService) you should design the interface exposed by each of these services.

/>

## Behavioral model

<**ToDo:**

* You may **break this section into subsections** to document the detailed behavioral model of each **major** component or use case.
* The behavioral aspects of your software components could be documented using:
* Activity diagrams to describe the overall flow of control of your application (step-by-step workflows of components in your system).
* Design Sequence Diagrams (DSD) to document how to realize key use cases. Method calls in the DSD should be numbered, and you show the parameters and the return type of each method.
* State diagrams for important parts of your system. You need to identify states and events that trigger state transitions.

/>

## Database design

<**ToDo:**

* Include a logical data model describing the major entities and the relationships between them. Also include the attributes for each entity. Make sure that the data model is consistent with (or derived from) the entities of the class diagram. If the model is too big partition the diagram into logical parts such as a diagram per component or subsystem.

/>

## User interface design

<**ToDo:**

* Discuss the target users for the system and any relevant characteristics that influenced the design decisions.
* Describe the overall layout and visual design of the user interface, including color schemes, typography, and graphics.
* Include screen mockups or wireframes for major use cases and explain the rationale behind key design decisions.
* Discuss the navigation design of the UI, including the structure of the user interface, movement between screens to achieve a particular use case, and any navigation aids or controls.
* Explain any novel aspects of your UI design and how they enhance the usability of your system. This could include innovative interactions, user feedback mechanisms, or features that support accessibility or personalization. />

## Design patterns

<**ToDo:**

* Document and evaluate the design patterns applied to your design such as the Model View Controller (MVC) pattern, the Factory pattern, the Proxy pattern, the Adapter etc.
* For each selected architectural pattern:
* Discuss the design rationale justifying the choice of the selected design pattern.
* Draw a UML diagram(s), e.g., class diagram, sequence diagram, to illustrate how the selected design pattern have been applied to your design.
* Evaluate of the used pattern the effect of selected pattern on your system quality attributes. Highlight the benefits introduced by the selected pattern potential problems or limitations introduced by the selected pattern.
* This subsection is extremely important and required. />

# Implementation

## Hardware/software used

< **ToDo:**

* List and discuss the hardware/software platforms and tools used for the design and implementation.
* Describe the role and purpose of each hardware and software component in the project. Explain why each component was chosen and how it fits into the overall project architecture.
* Provide references or links to relevant documentation, or resources for each hardware or software component.
* Use table format to present the above.

/>

Table 7. Hardware/software to be used

|  |  |
| --- | --- |
| HW/SW details | Justifications |
|  |  |
|  |  |

## Challenging issues and solutions

<**ToDo:**

* Discuss the challenging issues encountered during the implementation and how they were addressed. Lessons learnt from the experience that can be beneficial for others.

/>

# Testing and evaluation

<**ToDo:**

In this section, you should describe in detail the tests you have conducted to verify that your design and implementation satisfy the desired functional and non-functional requirements. The testing should verify and provide evidence that your solution solved the stated problem and satisfied the requirement specifications (if not, explain what is lacking).

Note that this section is a substantial portion of the grade for your final report, and it requires a significant effort.

/>

## Functional testing

<**ToDo:**

* Break this section into multiple subsections to test every use case.
* Testing should be done for different levels including at least **Unit Testing (a.k.a., component testing),** **Integration Testing, System Testing, and Acceptance Testing** to verify the correct delivery of the desired use cases documented in Section ‎3.2.
* Describe in detail the tests you have run to verify that your solution satisfies the functional requirements of your project. For example, for each use case, you should write test case(s) including the expected and actual results, run them and report the testing results. The test cases should be added to Appendix C. Functional testing will allow you to find errors/defects/faults/failures, then fix them and identify possible improvements. You need to have a comprehensive set of tests that verifies the correct functionality of every use case of your system.
* You should **present the test results**, with the appropriate level of detail in addition to accuracy and completeness**,** using tables, graphs, diagrams, screenshots, etc.
* **Discuss test results** and explain whether the implemented solution has satisfied the requirements. If not, state what is lacking or still needs improvement, then explain the reason for that.

/>

## Non-functional testing

<**ToDo:**

* Include a sub-section for each non-functional requirement listed in Section ‎3.3.
* Describe in detail the tests to **evaluate the quality of the system** (e.g., performance testing, usability testing, security testing, scalability testing, availability testing, etc. as documented in Section ‎3.3) to verify that your solution satisfies the non-functional requirements of your project. Include a summary table that shows each non-functional requirement and the measurements that prove your system met/did-not-meet the requirement. In case a requirement is not met, then explain the reason for that.
* You should **present the test results**, with an appropriate level of detail in addition to accuracy and completeness**,** using tables, graphs, diagrams, screenshots etc.
* **Discuss test results** and explain whether the implemented solution has achieved the requirements. If not, state what is lacking or still needs improvement, then explain the reason for that.

/>

# Conclusion

< **ToDo:**

* Discuss the main conclusions (e.g., match the project objectives with the achievements in your work and state the degree of achievement).
* Highlight the strengths of the solution and list down its shortcomings (what worked? what didn’t work?).
* Highlight the key contributions and the novel aspects of your work.

/>

# Future work

< **ToDo:**

* Suggested improvements and further work: identify areas of improvement in the project and features of interest that can be added later on. How the solution shortcomings could be addressed? What things could be done better? What additional resources are required to implement the extended / not-implemented design or features?

/>

# Student reflections

< **ToDo:**

* Add individual student reflections (add a sub-section for each student):
* Lessons learned from the project.
* Professional development you have achieved during your project experience (i.e., new skills gained) and explain its value for your future career.
* You can discuss (1) new technical skills acquired such as solving problems, designing and realizing solutions (2) interpersonal skills such as team leading and effective communication (3) personal growth such as adapting to change and acting professionally and ethically.
* Key shortcomings that you should avoid in future projects.
* Key lessons and new attitudes to carry forwards to your professional life from the personal experiences and the teamwork experienced during the project.

/>

# References

[1] C. T. Security, “Enterprise password security guidelines in a Nutshell: TechTarget,” Security, <https://www.techtarget.com/searchsecurity/post/Enterprise-password-security-guidelines-in-a-nutshell> (accessed Sep. 9, 2023).

[2] A. O’Driscoll, “25+ password statistics that may change your password habits,” Comparitech, <https://www.comparitech.com/blog/information-security/password-statistics/#:~:text=Another%202019%20Google%20study%20in,different%20password%20for%20every%20account>. (accessed Sep. 9, 2023).

[3] “Cost of a Data Breach Report”, IBM, <https://www.ibm.com/security/digital-assets/cost-data-breach-report/1Cost%20of%20a%20Data%20Breach%20Report%202020.pdf> ,2020 (accessed Sep. 9, 2023).

[4] “The human factor in Data Protection - Ponemon Institute,” Ponemon, <https://www.ponemon.org/local/upload/file/The_Human_Factor_in_data_Protection_WP_FINAL.pdf> (accessed Sep. 9, 2023).

# Appendix A – Use cases specification

< **ToDo:**

* Create a table for each use case. Use cases must be numbers e.g., UC01, UC02… See the ‘Use cases modeling’ slides for further details.

/>

|  |  |  |
| --- | --- | --- |
| **Use case Id:** UC**01** | <Use case Title> | |
| **Brief Description** |  | |
| **Primary actors** |  | |
| **Preconditions:** | | |
| **Post-conditions:** | | |
| **Main Success Scenario:**  <ToDo: List the included use cases. Add rows to the table below if needed /> | | |
| **Actor Action** | | **System Response** |
| 1. | | 2. |
|  | |  |
|  | |  |
|  | |  |
| **Alternative flows:**  ?.a. | | |
| **Special Requirements:**  <ToDo: List the non-functional requirements that the use case must meet> | | |

# Appendix B – Test cases specification

< **ToDo:**

* Create a table for each test case used to test the correct delivery of your functional requirements.

/>

**Test Case Template (Example)**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test case #** | 3.1 | **Associated user case ID** | U3 |
| Test designed by | Abbas Ibn Firnas | **Test design date** | 15/01/2023 |
| **Executed by** | Abbas Ibn Firnas | **Execution date** | 20/02/2023 |
| **Test case name** | ATM change PIN | | |
| **Short description** | Test the ATM change PIN use case | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Pre-conditions:**   * + The user has a valid ATM card   + The user has accessed the ATM by placing his ATM card in the machine   + The current PIN is 1234   + The system displays the main menu | | | | | |
| **Step** | **Test Step** | **Expected System Response** | **Actual Result** | **Pass/ Fail** | **Comment** |
| 1 | Click the 'Change PIN' button | The system displays a message asking the user to enter the new PIN | As expected | Pass |  |
| 2 | Enter '5555' | The system displays a message asking the user to confirm (re-enter) the new PIN | As expected | Pass |  |
| 3 | Re-enter '5555' | The system displays a message of successful operation  The system asks the user if he wants to perform other operations | As expected | Pass |  |
| 4 | Click 'YES' button | The system displays the main menu | As expected | Pass |  |
| 5 | Check post-condition 1 |  |  |  |  |
| 6 | Repeat steps 1,2,3 using another PIN say '6666' and click 'NO' button | The system is exited and displays a greeting message asking the user to place his ATM card in the machine |  |  |  |
| 7 | Check post-condition 2 |  |  |  |  |
| 8 | Repeat steps 1,2, using another PIN say '7777' | The system displays a message asking the user to confirm (re-enter) the new PIN |  |  |  |
| 9 | Enter a wrong confirmation (say ‘9876’) | The system displays a message of unsuccessful operation and asks the user to confirm the correct PIN |  |  |  |
| 10 | Re-enter ‘7777’ | The system displays a message of successful operation  The system asks the user if he wants to perform other operations |  |  |  |
| 11 | Click 'NO' button | The system is exited and displays a greeting message |  |  |  |
| 12 | Check post-condition 3 |  |  |  |  |
| **Post-conditions:**  1. The new PIN '5555' is saved in the database  2. The new PIN '6666' is saved in the database  3. The new PIN '7777' is saved in the database | | | | | |

# Other Appendices

< **ToDo:**

* The following are possible, additional appendices you may add to the project report:
* Any questionnaires, interview questions, etc. used in your project.
* Installation manual and deployment diagram
* User manual
* Any other appendices may be included to provide supporting details that could aid in understanding this report.

/>