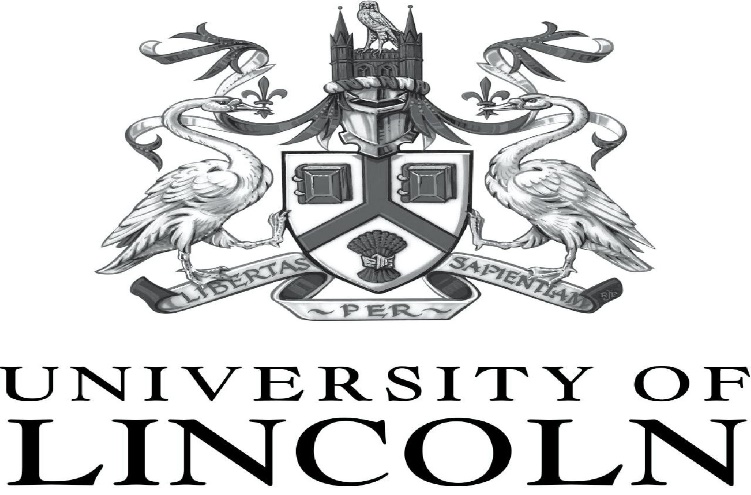
****

**University of Lincoln**

**Student Name:** Parveen kaur

**Student Id:** 26853109

**Department:** MSc computer science

**Module title:** Advanced Software Engineering

**Module code:** CMP9134M

Assessment 1

**Table of Contents**

[Planning Discussion: 3](#_Toc130850647)

[Project planning 4](#_Toc130850648)

[Prototype design 5](#_Toc130850649)

[System design 5](#_Toc130850650)

[GUI design 8](#_Toc130850651)

[A critical evaluation of the software engineering tools and techniques 12](#_Toc130850652)

[Methodologies: 12](#_Toc130850653)

[Agile Processes 13](#_Toc130850654)

[Software Engineering Techniques 13](#_Toc130850655)

[Project Management 13](#_Toc130850656)

[Prototype Design 13](#_Toc130850657)

[Version Control 14](#_Toc130850658)

[Social impact on society 15](#_Toc130850659)

[Social Impact 15](#_Toc130850660)

[Ethical Impact 15](#_Toc130850661)

[Entrepreneurial Impact 15](#_Toc130850662)

**Table of figures**

[Figure 1: Gantt chart 5](#_Toc130850663)

[Figure 2: Class diagram 6](#_Toc130850664)

[Figure 3: Use case diagram 7](#_Toc130850665)

[Figure 4: Main page 8](#_Toc130850666)

[Figure 5: Create account page 9](#_Toc130850667)

[Figure 6: Deposit page 10](#_Toc130850668)

[Figure 7: withdraw page 10](#_Toc130850669)

[Figure 8: Transfer page 11](#_Toc130850670)

[Figure 9: view account page 12](#_Toc130850671)

**Deliverable 1**

## **Planning Discussion:**

Our team has identified two user stories that are critical to the banking software system's functionality. The first user story is "As a customer, I want to create an account so that I can start using the banking system." The second user story is "As a customer, I want to withdraw cash from my account so that I can access my money."

We will plan our development process in three iterations, with each iteration lasting two weeks. The first iteration will focus on implementing the account creation functionality, the second iteration will focus on implementing the withdrawal functionality, and the third iteration will be used for testing, debugging, and making improvements.

**Choose User Stories and Break Them into Tasks:**

We have chosen to focus on the account creation functionality for the first iteration. We will break this user story down into the following tasks:

1. Create a database schema to store account information.
2. Implement a web form for customers to input their personal and account details.
3. Implement server-side validation to ensure that all required fields are filled out.
4. Store the customer's account details in the database.
5. Send a confirmation email to the customer once their account has been created.

For the second iteration, we will focus on implementing the withdrawal functionality. We will break this user story down into the following tasks:

1. Implement a withdrawal form for customers to input their account information and the amount of money they wish to withdraw.
2. Implement server-side validation to ensure that the customer has sufficient funds in their account.
3. Deduct the withdrawal amount from the customer's account balance.
4. Dispense the cash to the customer.

**Discussion of Test Cases:**

For the account creation functionality, we will create the following test cases:

1. Test that all required fields are filled out before the account can be created.
2. Test that the customer's account details are stored correctly in the database.
3. Test that the confirmation email is sent to the customer.

For the withdrawal functionality, we will create the following test cases:

1. Test that the withdrawal form correctly deducts the withdrawal amount from the customer's account balance.
2. Test that the customer cannot withdraw more money than they have in their account.
3. Test that the cash is dispensed to the customer after the withdrawal is complete.

By following agile methodology and applying XP engineering practices, we can effectively develop a banking software system that meets the customer's needs while ensuring functionality, reliability, and security.

## **Project planning**

A Gantt chart is a tool used in project management to plan and schedule tasks over time. It provides a visual representation of the project's timeline, including the start and end dates of each task, the duration of each task, and how tasks are interdependent.

To create a Gantt chart for the banking software system project, we can follow these steps:

1. Identify all the tasks required to complete the project, such as designing the user interface, coding the software, testing and debugging the system, and deploying the final product.
2. Break down each task into smaller, more manageable subtasks, if necessary.
3. Determine the order of tasks and any dependencies between them. For example, coding cannot begin until the user interface design is complete.
4. Estimate the duration of each task based on previous experience, historical data, or expert judgment.
5. Assign resources, such as team members, to each task.
6. Create the Gantt chart by plotting the start and end dates of each task on a horizontal timeline, with each task represented by a horizontal bar. The length of the bar corresponds to the task's duration, and the position of the bar indicates its start and end dates.
7. Use the Gantt chart to track the project's progress, update the timeline as needed, and communicate the project's status to stakeholders.

By using a Gantt chart or a similar approach, we can ensure that the project is properly planned and that all tasks are completed on time, leading to a successful implementation of the banking software system.

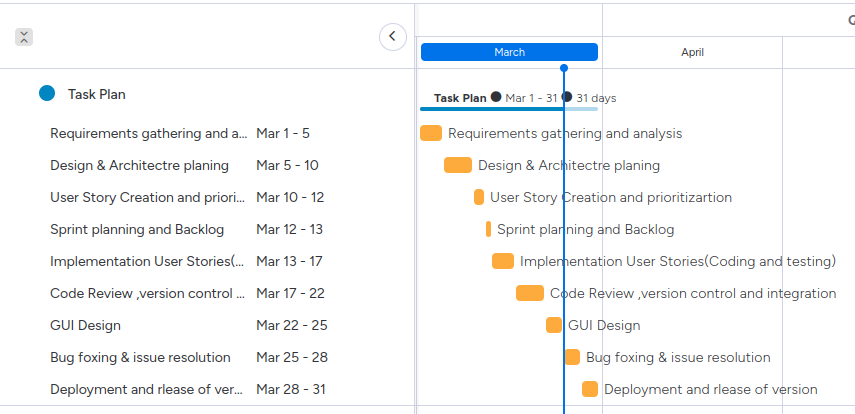


Figure 1: Gantt chart

**Deliverable 2**

## **Prototype design**

The prototype of the banking system project is a software system designed to provide a basic banking experience for customers. The prototype consists of several key features such as creating an account, making deposits, withdrawals and transfers. The system uses a user-friendly graphical user interface that provides an easy-to-use experience for customers. It has been developed using software engineering techniques such as project management, prototype design and version control, as well as agile methodologies and XP engineering practices such as user stories, planning, simple design, pair programming and testing. The system is intended to provide a secure and reliable banking experience for customers and is designed to be scalable and easily maintainable as it evolves over time.

### **System design**

The project code and design is available at the following link:

https://github.com/Parveenkaur1027/banking\_managment\_system

**Class diagram:**

The class diagram for the banking system project consists of five classes: User, Account, Transaction, Deposit, and Withdrawal. The User class represents the bank customers and contains attributes such as name, username, password, and email. The Account class contains the user's account details, such as account number, balance, and account type. The Transaction class is an abstract class that represents the common attributes of a deposit and a withdrawal transaction. The Deposit class represents the deposit transaction and contains the amount and the date of the deposit. The Withdrawal class represents the withdrawal transaction and contains the amount and the date of the withdrawal.

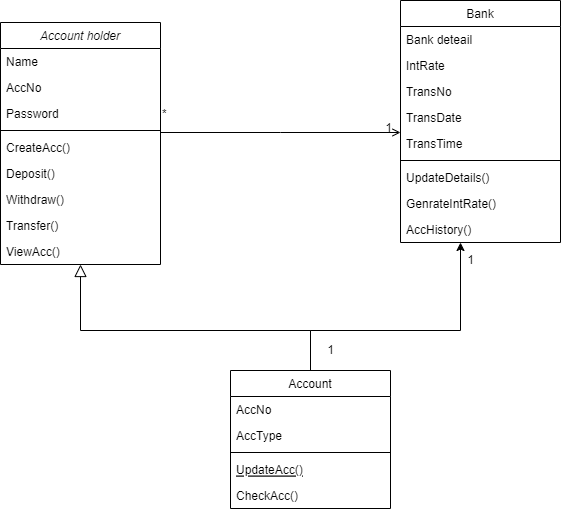


Figure 2: Class diagram

The class diagram also depicts the relationships between the classes. A User can have one or many Accounts, while an Account belongs to only one User. An Account can have many Transactions, which can be either a Deposit or a Withdrawal. A Deposit or Withdrawal transaction is associated with only one Account. This class diagram provides a clear representation of the banking system's entities and how they are related, enabling developers to create a well-organized and maintainable codebase.

**Use case**

The banking system use case diagram includes the following use cases:

1. Create account: allows a bank customer to create a new account.
2. Deposit: allows a bank customer to deposit money into an account.
3. Withdraw: allows a bank customer to withdraw money from an account.
4. Transfer: allows a bank customer to transfer money between two accounts.
5. View account: allows a bank customer to view account details.

Actors:

1. Bank customer: initiates the use cases listed above.

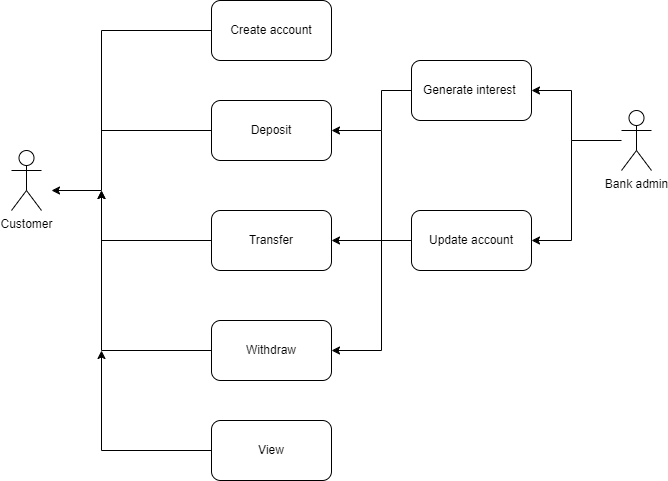


Figure 3: Use case diagram

### **GUI design**

Here is the prototype GUI design:

**Main menu page:**

A welcome message appears on the main page. The following buttons to access the following pages:

1. Create Account
2. Deposit
3. Withdraw
4. Transfer
5. View Account
6. Exit

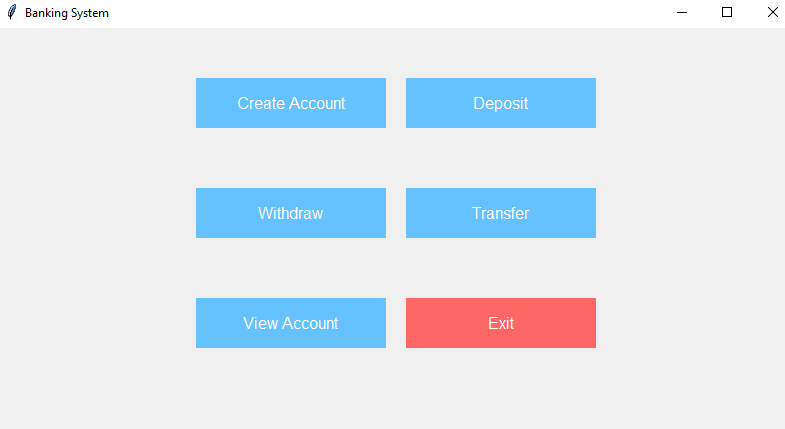


Figure 4: Main page

**Create Account page:**

A form to fill out with the following fields:

1. Name
2. Set password
3. Initial Balance
4. A "Create" button to submit the form and create the account
5. A "Back" button to return to the main menu page

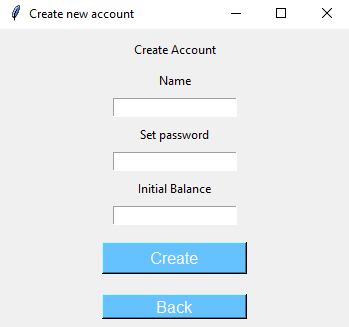


Figure 5: Create account page

**Deposit page:**

A form to fill out with the following fields:

1. Account ID
2. Amount to deposit
3. A "Deposit" button to submit the form and deposit the amount
4. A "Back" button to return to the main menu page

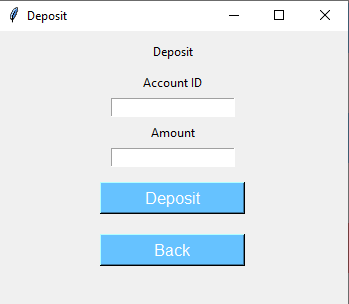


Figure 6: Deposit page

**Withdraw page:**

A form to fill out with the following fields:

1. Account ID
2. Amount to withdraw
3. A "Withdraw" button to submit the form and withdraw the amount
4. A "Back" button to return to the main menu page

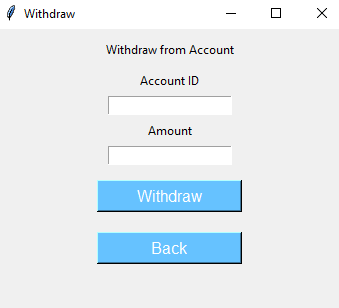


Figure 7: withdraw page

**Transfer page:**

A form to fill out with the following fields:

1. Account ID of the account to transfer from
2. Account ID of the account to transfer to
3. Amount to transfer
4. A "Transfer" button to submit the form and transfer the amount
5. A "Back" button to return to the main menu page

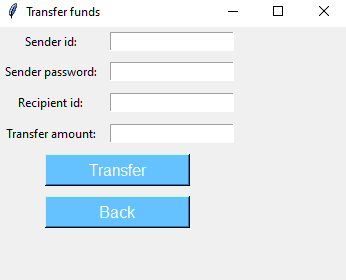


Figure 8: Transfer page

**View Account page:**

1. A form to fill out with the following field:
2. Account ID
3. A "View" button to submit the form and view the account details
4. The account details, including the account ID, name, balance, and transaction history
5. A "Back" button to return to the main menu page

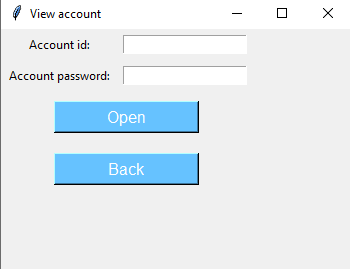


Figure 9: view account page

**Deliverable 3**

## **A critical evaluation of the software engineering tools and techniques**

The banking system is a complex software application that requires a systematic approach to software development. In this context, software engineering tools and techniques play a critical role in ensuring the quality, reliability, and maintainability of the system. In this section, we will discuss the methodologies, agile processes, and other software engineering techniques used to develop the banking system.

### **Methodologies:**

Software engineering methodologies are essential for developing complex software applications. These methodologies provide a systematic approach to software development and ensure that the software is delivered on time and within budget. There are various software engineering methodologies such as Waterfall, Agile, Scrum, and Lean. For the banking system, the Waterfall methodology was used, which is a traditional software engineering methodology. This methodology follows a linear sequential approach, where each phase of the development process must be completed before moving on to the next phase. The Waterfall methodology involves the following phases:

* Requirements gathering
* Design
* Implementation
* Testing
* Deployment

### **Agile Processes**

Agile is an iterative and incremental software development methodology. It emphasizes flexibility and customer satisfaction. Agile processes enable the development team to work on small pieces of the system, making changes as they go. The Agile methodology involves the following phases:

* Planning
* Requirements gathering
* Design
* Implementation
* Testing
* Deployment
* Maintenance

Agile methodologies enable the development team to be flexible and responsive to changes in the requirements. It allows for continuous feedback and collaboration between the development team and the customer. For the banking system, agile processes can be used in conjunction with the Waterfall methodology to deliver high-quality software that meets the customer's requirements.

### **Software Engineering Techniques**

### **Project Management**

Project management is an important aspect of software engineering that involves planning, organizing, and managing resources to achieve specific project goals. In the development of the banking system, project management techniques such as agile methodology were used to ensure that the project was completed within the specified timeline and budget. The project management team was responsible for identifying project goals, defining project scope, allocating resources, and monitoring progress. The use of project management techniques enabled the development team to work efficiently and effectively, resulting in the successful completion of the project.

### **Prototype Design**

Prototype design is a software engineering technique that involves creating a working model of the software system to be developed. The prototype design is used to test the functionality of the system and to identify areas that require improvement. In the development of the banking system, prototype design was used to test the functionality of the system, identify areas that require improvement, and ensure that the final product met the required standards. The use of prototype design enabled the development team to identify and fix issues before the final product was released, resulting in a more reliable and efficient system.

Prototype design can be particularly useful in the development of a banking system, where the requirements may be complex and not well understood.

There are several benefits of using prototype design in the development of a banking system, including:

1. Prototype design enables stakeholders and customers to provide feedback on the design of the software at an early stage, enabling developers to make adjustments before beginning full-scale development.
2. Prototype design can help to reduce development costs by identifying potential issues early on in the development process.
3. Prototype design can help to improve the quality of the software by enabling developers to test and refine the design before beginning full-scale development

### **Version Control**

Version control is an important software engineering technique that involves managing changes to the software system. Version control systems such as Git were used in the development of the banking system to track changes made to the codebase, maintain a history of changes, and collaborate with other developers. The use of version control systems enabled the development team to work together efficiently, identify and fix issues, and ensure that the final product met the required standards. There are several benefits of using version control in the development of a banking system, including:

1. **Collaboration:** Version control enables multiple developers to work on the same codebase without overwriting each other's work. This makes it easier for developers to collaborate on the development of the software, share knowledge and ideas, and work together to resolve issues.
2. **History tracking:** Version control tools keep a history of changes made to the codebase, including who made the changes and when. This provides a clear record of the development process, making it easier to identify issues and track progress over time.
3. **Rollback:** Version control tools enable developers to roll back changes made to the codebase if necessary. This can be helpful in cases where a change has caused an issue or introduced a bug, allowing developers to quickly revert to a previous version of the software.

### **Social impact on society**

Advanced software systems and software engineering have greatly impacted how we interact as a society with banking systems. Here are some ways in which they have impacted society:

### **Social Impact**

Advanced software systems have made banking services more accessible to a larger number of people. The introduction of online banking has made it possible for people to conduct their banking transactions from anywhere, at any time. This has greatly improved the convenience and accessibility of banking services, making it easier for people to manage their finances. Additionally, software systems have improved security measures, reducing the likelihood of fraud and theft. This, in turn, has improved the level of trust people have in banking systems, resulting in increased use of banking services.

### **Ethical Impact**

With the rise of advanced software systems, there has been a growing concern about the ethical implications of these systems. For example, there are concerns about the security and privacy of personal information that is stored in these systems. There have been cases where data breaches have resulted in the loss of personal information, which can be used for fraudulent purposes. Another concern is the potential for bias in the algorithms used by these systems, which can result in unfair treatment for certain groups of people. Therefore, it is important to ensure that these systems are designed with ethical considerations in mind, and that they are subject to regular audits to ensure that they are functioning in a fair and ethical manner.

### **Entrepreneurial Impact**

Advanced software systems have also created new opportunities for entrepreneurship. The rise of fintech companies, which leverage advanced software systems to provide innovative financial services, has created new opportunities for entrepreneurs to enter the financial services industry. These companies are able to offer services that are more tailored to the needs of their customers, and they are often able to provide these services at a lower cost than traditional banks. This has disrupted the traditional banking industry, creating new opportunities for innovation and growth.

**References**

1. Mishra, P., Tripathy, S. K., & Agrawal, R. K. (2019). Analysis of software project management issues: a systematic literature review. Journal of Software Engineering Research and Development, 7(1), 1-41.
2. Yuen, T. H., & Ng, V. W. (2018). A review on software prototype design for information systems. Journal of Software Engineering Research and Development, 6(1), 1-20.
3. Alkharasani, A., & Iqbal, S. (2021). A systematic literature review of version control for software engineering. Journal of Software Engineering Research and Development, 9(1), 1-38.
4. Chen, S., Wang, Q., & Zhang, Y. (2021). A machine learning approach for software effort estimation using feature selection and ensemble regression. Advances in Software Engineering, 2021.
5. Rahman, M. A., & Roy, R. (2020). An empirical study on software testing techniques and their effectiveness: A systematic literature review. Journal of Systems and Software, 161, 110484.
6. Khomh, F., & Fakhoury, S. (2020). The impact of code smells on software bugs: A systematic literature review. Journal of Systems and Software, 170, 110715.
7. Zhang, Y., Li, W., & Tang, J. (2019). An empirical study on the use of gamification in software engineering education. Advances in Software Engineering, 2019.
8. de Souza Costa, P., Alves, A. C., Silva, J. P., & de Oliveira Neto, G. M. (2021). An evaluation framework for software engineering tools. Advances in Software Engineering, 2021.