|  |
| --- |
| **Batch: A2 Roll No.: 1811037**  **Experiment No.: 1**  **Grade : AA/AB/BB/BC/CC/CD/DD**  **Signature of the Staff In-charge with date :** |

Group:

1811024 Yash Mehta

1811028 Nidhi Nair

1811037 Om Rawal

|  |
| --- |
| **Title: Prepare documents related to your mini project** |

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Objective:** Prepare roadmap for developing a Mini project **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Expected Outcome of Experiment:**

|  |  |
| --- | --- |
|  | **At the end of successful completion of the course, the student will be able to** |
| CO1 | Define the problem statement and scope of the problem |
| CO2 | Identify various hardware and software requirements for problem solution |
| CO5 | Prepare a technical report based on the Mini project. |

**Introduction**:

We use the drive to store our important documents, records, images as well as memos. So it is a repository where the user can store, view, edit as well as download their files as and when required. But the major disadvantage of these drives is that there is always a sole administrator which has access to all the accounts. So in case of any suspicious or malicious activity, he/she(admin) can have a look at our account. This enhanced us to engender an idea to implement a secure drive that will value the highly rated motto of furtiveness as well as privacy which is the most desired principle in today’s era.

**The objective of the project:**

Overcome the problem of the Administrator’s access to the contents of the Database even though the contents are restricted to the authenticated user. Hence to overcome this issue the files will be stored in an encrypted format and the contents will not be visible to the Administrator. Also to keep the files secured from unauthorized users or any malicious activity, the Drive will be immune to attacks and penetrations using encryption and two-step authentication.

**Scope of the project:**

* + Providing Authentication and Encryption.
  + Connectivity between the contents of the database and the hosting server of the database for successful transactions.
  + Maintain the system and fix the bugs and upgrade up to ONE month after commissioning.
  + Procurement of hardware for using the system is **OUT OF SCOPE** of this project.

**Requirements gathering:**

**Functional Requirements:**

1. The system shall allow a user to Store all documents/files.
2. The system shall allow a user to view his/her Stored documents.
3. The system shall allow the user to update his password.
4. The system shall allow a user to update his Picture for recognition.
5. The system shall store the files in a database in an encrypted format.
6. The system shall display all the stored files of the user.
7. The system shall provide a user with only 3 unsuccessful attempts of login.
8. The system shall accept an image of the user only from the webcam to assure the presence of the user.
9. The system shall upload files less than 100Mb at an initial stage.

**Non-Functional Requirements:**

1. Frames rate

The minimum frame rate must be twenty frames per second. The average frame rate must be greater than 30. The frame rate can be monitor directly from the graphic engine.

2. Response time

The average response time between click and reaction must be less than 0.5 seconds. The maximum response time between click and reaction must be two seconds. Adding some simple classes and methods that will compute and display the time needed to process any operation can test these requirements.

3. Required resources

The file can be any size ranging from a few MB to GB. The Software must use less than 50MB of disk space.

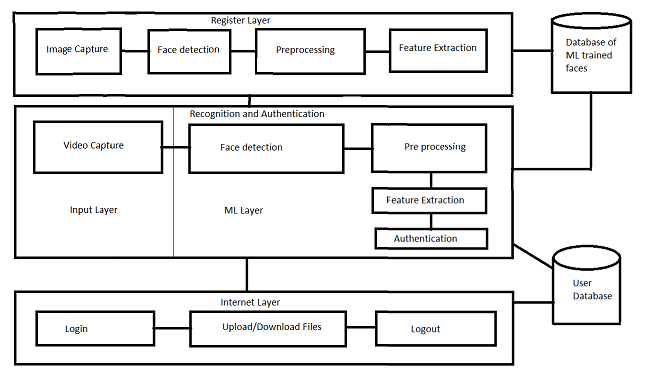
4. Platform

The game must run on all servers.

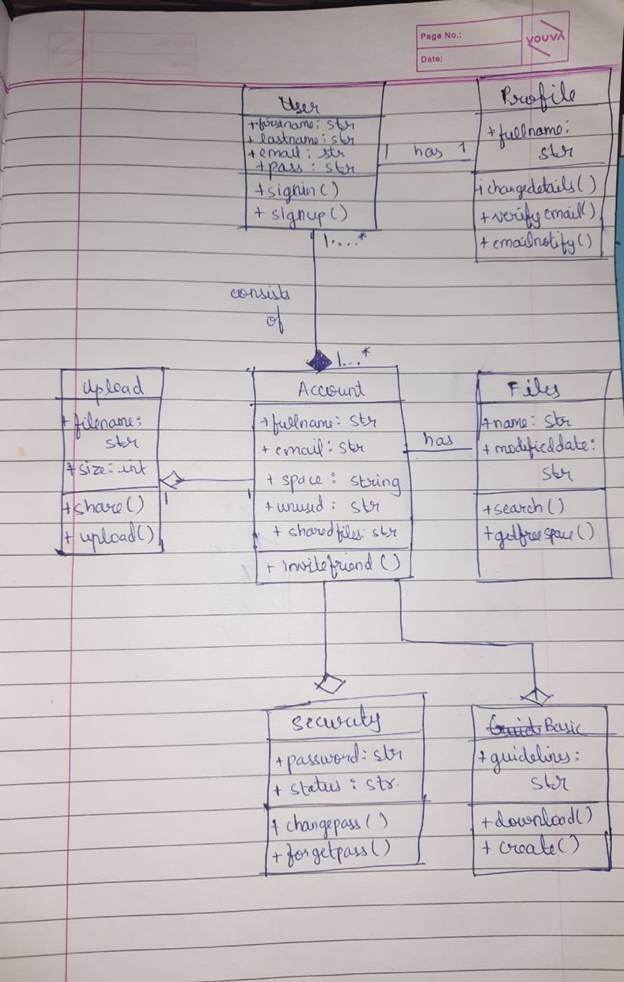
5. Maintainability

The code written for the software must be maintainable. Also adding documentation will improve the maintainability scale of the system.

**Design:**

****

Class Diagram for the secure drive:



**Plan:**

Planning is very essential for the successful completion of any activity in which multiple stakeholders are involved. To start with one will write down all activities needed to be carried out mentioning the role and responsibility of each human resource. This will also help in sequencing and tracking the progress of the development process. A sample Role and Responsibility matrix could be as follows, Please prepare according to the needs of your project.

|  |  |  |  |
| --- | --- | --- | --- |
| Activity | YASH | NIDHI | OM |
| **1.** **Requirement Gathering** |  |  |  |
| 1.1 Interaction with customer | C | C | C |
| 1.2 Preparing SRS | C | C | C |
| **2.** **Design** |  |  |  |
| 2.1 Preparing Block diagram | C | R | A |
| 2.2 Writing Functional Requirements | R | C | R |
| 2.3 Writing Non-Functional Requirements | R | C | R |
| 2.4 Developing Use Case | C | A | A |
| 2.5 Developing Test Cases | R | C | C |
| 1. **Planning** | R | C | C |
| 1. **Coding** |  |  |  |
| 4.1 Player | C | C | C |
| 4.2 Procedural Generation | C | C | C |
| 4.3 Rhythm Group | C | C | C |
| 4.4 Front end/ UI | C | C | C |
| 1. **Testing** |  |  |  |
| 5.1 Player | E | A | E |
| 5.2 Procedural Generation | A | E | E |
| 5.3 Rhythm Group | E | A | E |
| 5.4 Front End/UI | A | E | R |
| 5.4 System Testing | E | E | A |

C: Creator, R: Reviewer, A: Approver E: Executor

TIMELINE CHART -

<https://docs.google.com/spreadsheets/d/16JoXlSevzRgdoHFiGv-YFfYpDjN32XEg/edit#gid=282162638>

PLEASE REFER THIS LINK TO VIEW OUR TIMELINE (GANTT) CHART FOR THE MINI PROJECT.

**Testing:**

1. **Unit Testing.**

We plan to test all the modules individually first. We will implement each module and test it before integrating it with others. We will test whether Face recognition authentication and file uploading are working as expected, check if the encryption process of the files is not buggy, and make sure that the files are first encrypted while storing and then decrypted while downloading.

1. **Integration Testing**

We will integrate all the modules and test their compatibility and accordingly make changes to key variables that affect the software. First, the face recognition and authentication module will be integrated, followed by the UI of drive and upload as well as download functionalities, and finally the encryption-decryption module.

**3)**  **System Testing**

After the project is ready, in system testing we test for the working of the entire system and look for bugs and solve them.

**Conclusion:**

We have successfully implemented the various phases of the software development cycle. We have also created the requirement specifications, design document, project plan, and testing plan for our project.

**Post Lab Activities (with reference to your tool):**

**1. What are various Static & Dynamic Analysis Models?**

**ANSWER:**

Static analysis involves a set of methods used to analyze software source code or object code, determine how the software functions, and establish criteria to check its correctness. Static analysis studies the source code without executing it and reveals a wide variety of information such as the structure of the model used, data and control flow, syntax accuracy, and more.

There are several types of static analysis methods-

1. Control Analysis: -

This software focuses on examining the controls used in calling structure, control flow analysis, and state transition analysis. The calling structure is related to the model by identifying the calling and call structure. The calling structure can be a process, subroutine, function, or method. Control flow analysis checks the sequence of control transfers. Furthermore, inefficient constructions in the model. A graph of the model is created in which the conditional branches and model junctions are represented by nodes.

2. Data Analysis: -

Ensures proper operation is applied to data objects such as data structures and linked lists. Also, this method ensures that the defined data is used properly. Data analysis involves two methods, namely, data dependency and data-flow analysis. Data dependency is necessary to assess the accuracy of synchronization across multiple processors. Data flow analysis checks the definition and context of variables.

3. Fault/Failure Analysis: -

It analyses faults (incorrectly component) and failure (incorrect behavior of model component) in the model. This method uses the input-output transformation description to identify the conditions that are cause for the failure. To determine the failures in certain conditions the model design specification is checked.

4. Interface Analysis: -

This software verifies and verifies interactive and distribution simulations to check the code. There are two basic techniques for interface analysis and user interface analysis examines sub-model interfaces and determines the accuracy of the interface structure. User interface analysis examines the user interface model and for the precautionary steps taken to prevent errors during the user’s interaction with the model. This method also focuses on how accurately the interface is integrated into the overall model and simulation.

Dynamic analysis is the analysis of computer software that is performed by executing programs on a real or virtual processor. For dynamic analysis to be effective, the target program must be executed with sufficient test inputs to cover almost all possible outputs.

1. Code coverage

Computing the code coverage according to a test suite or a workload is a standard dynamic analysis technique.

· Gcov is the GNU source code coverage program.

· VB Watch injects dynamic analysis code into Visual Basic programs to monitor code coverage, call stack, execution trace, instantiated objects, and variables.

2. Memory error detection

· Intel Inspector: Dynamic memory error debugger for C, C++, and Fortran applications that run on Windows\* and Linux\*.

· AddressSanitizer: Memory error detection for Linux, macOS, Windows, and more. Part of LLVM.

· BoundsChecker: Memory error detection for Windows-based applications. Part of Micro Focus DevPartner.

· Dmalloc, a library for checking memory allocation and leaks. The software must be recompiled, and all files must include the special C header file malloc. h.

3. Fault localization

Fault localization refers to locating the buggy code (for example the buggy statement) according to failing and passing test cases. For example, Tarantula is a well-known fault localization approach based on the covered code. Fault localization illustrates an important property of dynamic analysis: the results of the analysis depend on the considered workload, inputs, or test cases. For fault localization, it has been shown that one can refactor the test cases to get better results.

4. Invariant inference

Daikon is an implementation of dynamic invariant detection. Daikon runs a program, observes the values that the program computes, and then reports properties that were true over the observed executions, and thus likely true over all executions.

5. Security analysis

Dynamic analysis can be used to detect security problems.

· IBM Rational AppScan is a suite of application security solutions targeted for different stages of the development lifecycle. The suite includes two main dynamic analysis products - IBM Rational AppScan Standard Edition, and IBM Rational AppScan Enterprise Edition. Also, the suite includes IBM Rational AppScan Source Edition - a static analysis tool.

**2. What are the merits and limitations of Component-based/ Pattern Based design approaches?**

**ANSWERS:**

Component-based architecture focuses on the decomposition of the design into individual functional or logical components that represent well-defined communication interfaces containing methods, events, and properties. It provides a higher level of abstraction and divides the problem into sub-problems, each associated with component partitions.

MERITS:

1. Reduced time in the market and the development cost by reusing existing components.

2. Ease of deployment − As new compatible versions become available, it is easier to replace existing versions with no impact on the other components or the system.

3. Reduced cost − The use of third-party components allows you to spread the cost of development and maintenance.

4. Ease of development − Components implement well-known interfaces to provide defined functionality, allowing development without impacting other parts of the system.

5. Reusable − The use of reusable components means that they can be used to spread the development and maintenance cost across several applications or systems.

6. Modification of technical complexity − A component modifies the complexity using a component container and its services.

7. Reliability − The overall system reliability increases since the reliability of each component enhances the reliability of the whole system via reuse.

8. System maintenance and evolution − Easy to change and update the implementation without affecting the rest of the system.

9. Independent − Independence and flexible connectivity of components. Independent development of components by different groups in parallel. Productivity for software development and future software development.

Limitations:

1. Increased development time

2. Difficult to identify requirements

3. Decreases usability

4. Choosing middleware is difficult due to different components being built independently Technologies may be incompatible

5. Testing becomes hard as there may be unknown uses of a component

**3. What are the attributes of good software?**

**ANSWERS:**

The three characteristics of good application software are:-

1) Operational Characteristics

2) Transition Characteristics

3) Revision Characteristics

What Operational Characteristics should a software have?

These are functionality based factors and related to the 'exterior quality' of software. Various Operational Characteristics of software are:

a) Correctness: The software which we are making should meet all the specifications stated by the customer.

b) Usability/Learnability: The number of efforts or time required to learn how to use the software should be less. This makes the software user-friendly even for IT-illiterate people.

c) Integrity: Just like medicines have side-effects, in the same way, the software may have a side-effect i.e. it may affect the working of another application. But a quality software should not have side effects.

d) Reliability: The software product should not have any defects. Not only this, it shouldn't fail while execution.

e) Efficiency: This characteristic relates to the way software uses the available resources. The software should make effective use of the storage space and execute commands as per desired timing requirements.

f) Security: With the increase in security threats nowadays, this factor is gaining importance. The software shouldn't have ill effects on data/hardware. Proper measures should be taken to keep data secure from external threats.

g) Safety: The software should not be hazardous to the environment/life.

**4.What are the Revision Characteristics of software?**

These engineering based factors relate to the 'interior quality' of the software like efficiency, documentation, and structure. These factors should be in-build in any good software.

Various Revision Characteristics of software are: -

a) Maintainability: Maintenance of the software should be easy for any kind of user.

b) Flexibility: Changes in the software should be easy to make.

c) Extensibility: It should be easy to increase the functions performed by it.

d) Scalability: It should be very easy to upgrade it for more work (or for more number of users).

e) Testability: Testing the software should be easy.

f) Modularity: Any software is said to be made of units and modules which are independent of each other. These modules are then integrated to make the final software. If the software is divided into separate independent parts that can be modified, tested separately, it has high modularity.

Transition Characteristics of the software:

a) Interoperability: Interoperability is the ability of software to exchange information with other applications and make use of information transparently.

b) Reusability: If we can use the software code with some modifications for different purposes then we call software to be reusable.

c) Portability: The ability of software to perform the same functions across all environments and platforms demonstrates its portability.