Secure Backup Software System
Analysis and Design Document

Student: Ciucescu Vlad Andrei
Group: 30432

Revision History

Date	Version	Description	Author
04/Apr/18	1.0	First version of the project analysis and design document	Ciucescu Vlad
19/Apr/18	1.1	Added domain model, updated conceptual architecture, package diagram, deployment diagram, component diagram	Ciucescu Vlad
21/Apr/18	1.2	Added design model, data model and testing methods	Ciucescu Vlad

Table of Contents

I. Project Specification 4

II. Elaboration – Iteration 1.1 4

1. Domain Model 4

2. Architectural Design 5

2.1 Conceptual Architecture 5
2.2 Package Design 6
2.3 Component and Deployment Diagrams 6

III. Elaboration – Iteration 1.2 7

1. Design Model 7

1.1 Dynamic Behavior 7
1.2 Class Design 8

2. Data Model 9

3. Unit Testing 10

IV. Elaboration – Iteration 2 10

1. Architectural Design Refinement 10

2. Design Model Refinement 10

V. Construction and Transition 10

1. System Testing 10

2. Future improvements 10

VI. Bibliography 11

I. Project Specification

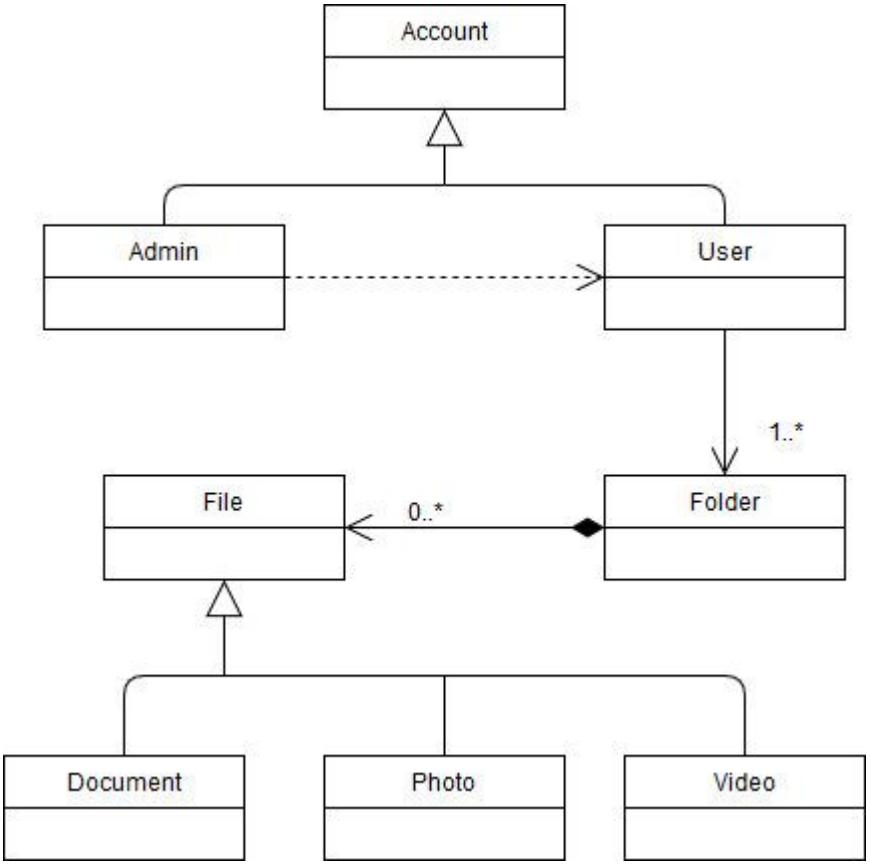
A software backup system should be implemented, which allows users to store files, documents, images and videos through a windows desktop application in a secure manner. A separate folder should be created for each user for them to store their files in. This folder should be only accessible to authorized users. All the details of the users, user activity and files will be stored in a RDBMS.

There will exist two types of users:

- Administrator – has the responsibility of managing user accounts (creating new accounts, deleting accounts at the request of users, blocking a user etc.)
- Regular User – can use the software to store and retrieve files. Can access their account after an administrator has created it and can request an administrator to close their account.

II. Elaboration – Iteration 1.1

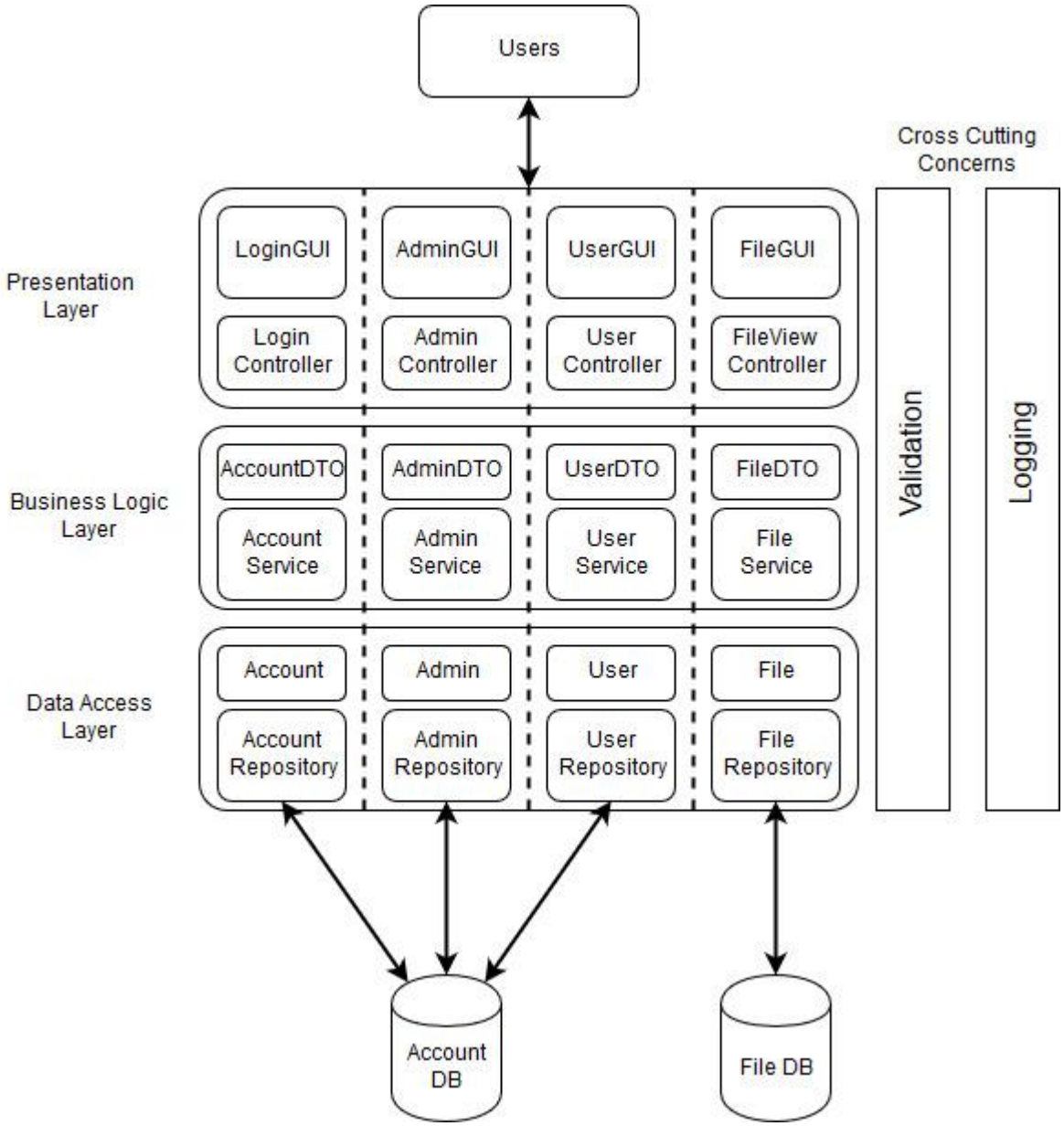
1. Domain Model



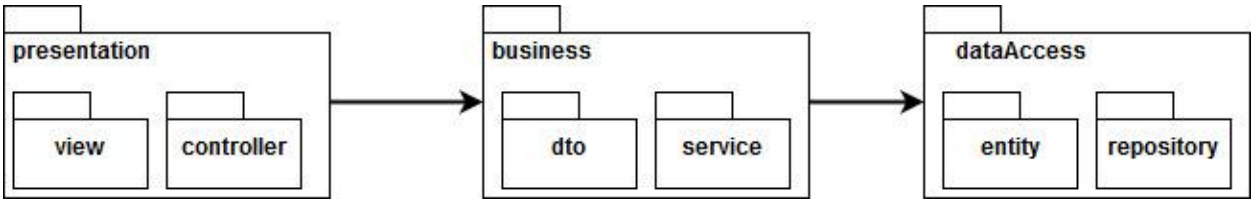
2. Architectural Design

2.1 Conceptual Architecture

The proposed architecture of the application is the layers architecture. This aids in separating the responsibilities, because each layer is designed to perform a specific role within the application. As such, each layer will be cohesive and easily replaceable, should the need arise. This system will be composed of three layers: data access layer, business logic layer and presentation layer.

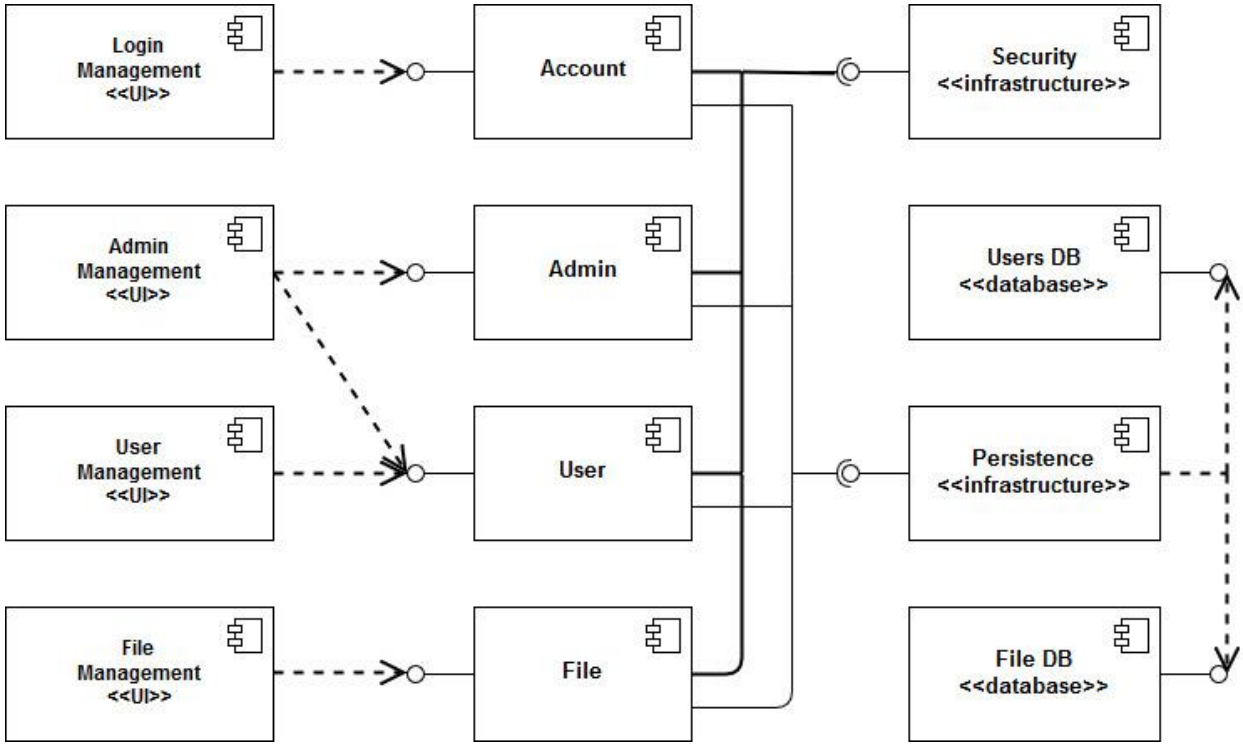


2.2 Package Design

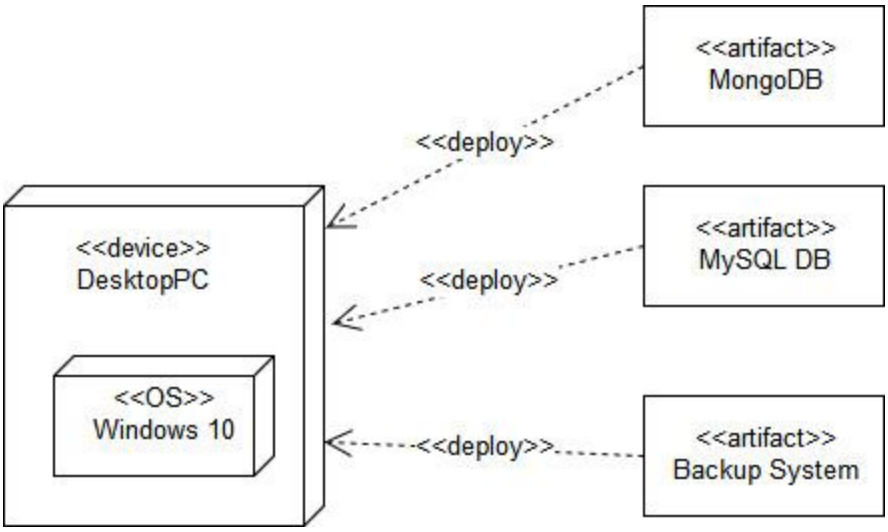


2.3 Component and Deployment Diagrams

Component Diagram



Deployment Diagram

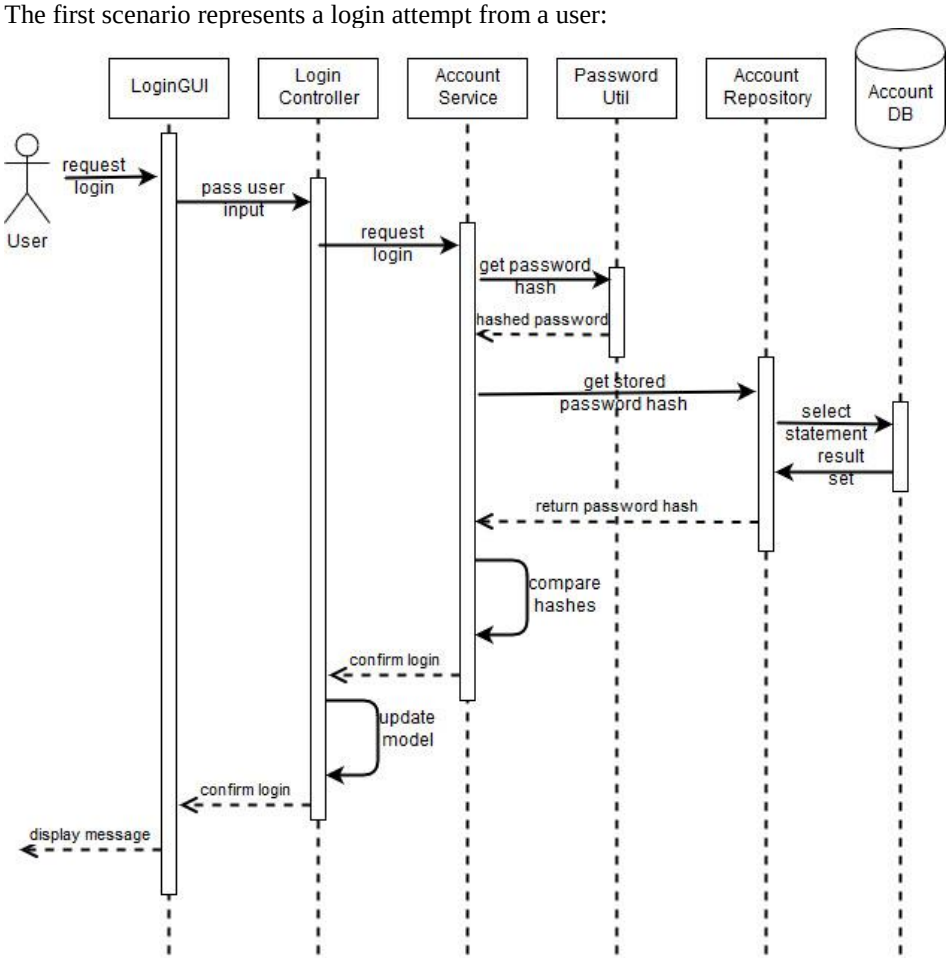


III. Elaboration – Iteration 1.2

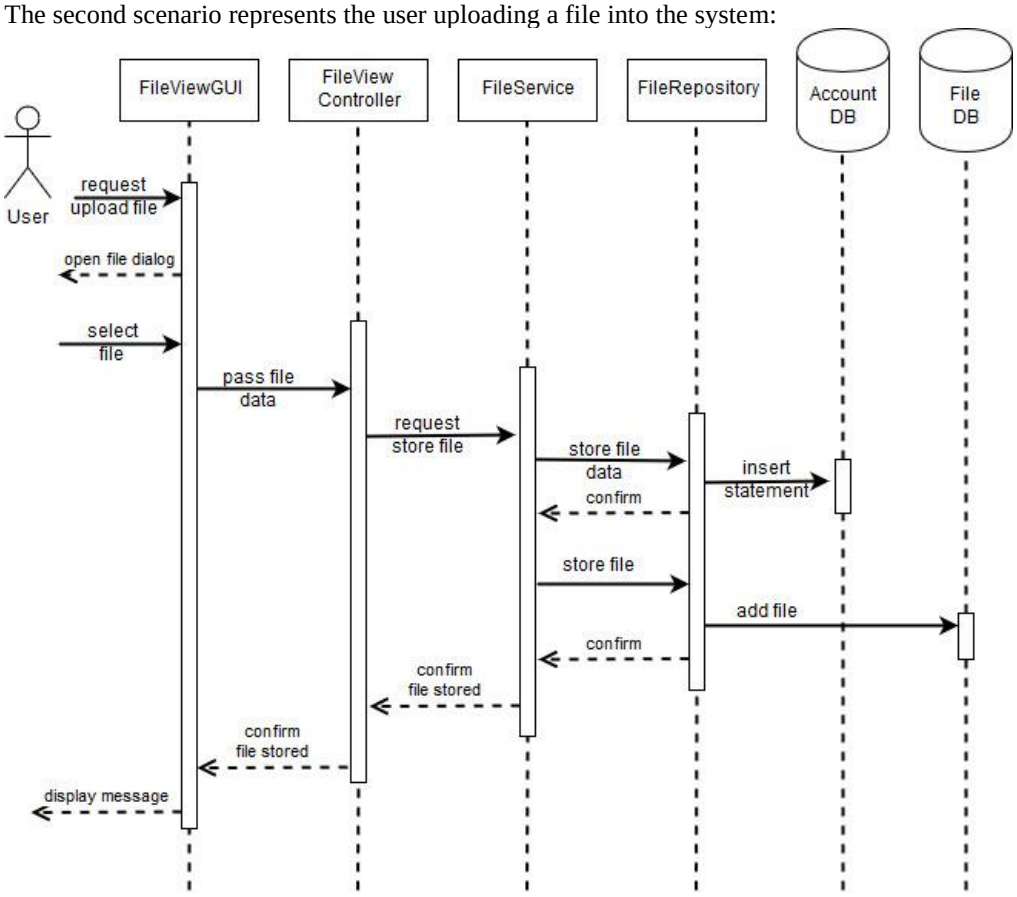
1. Design Model

1.1 Dynamic Behavior

The first scenario represents a login attempt from a user:



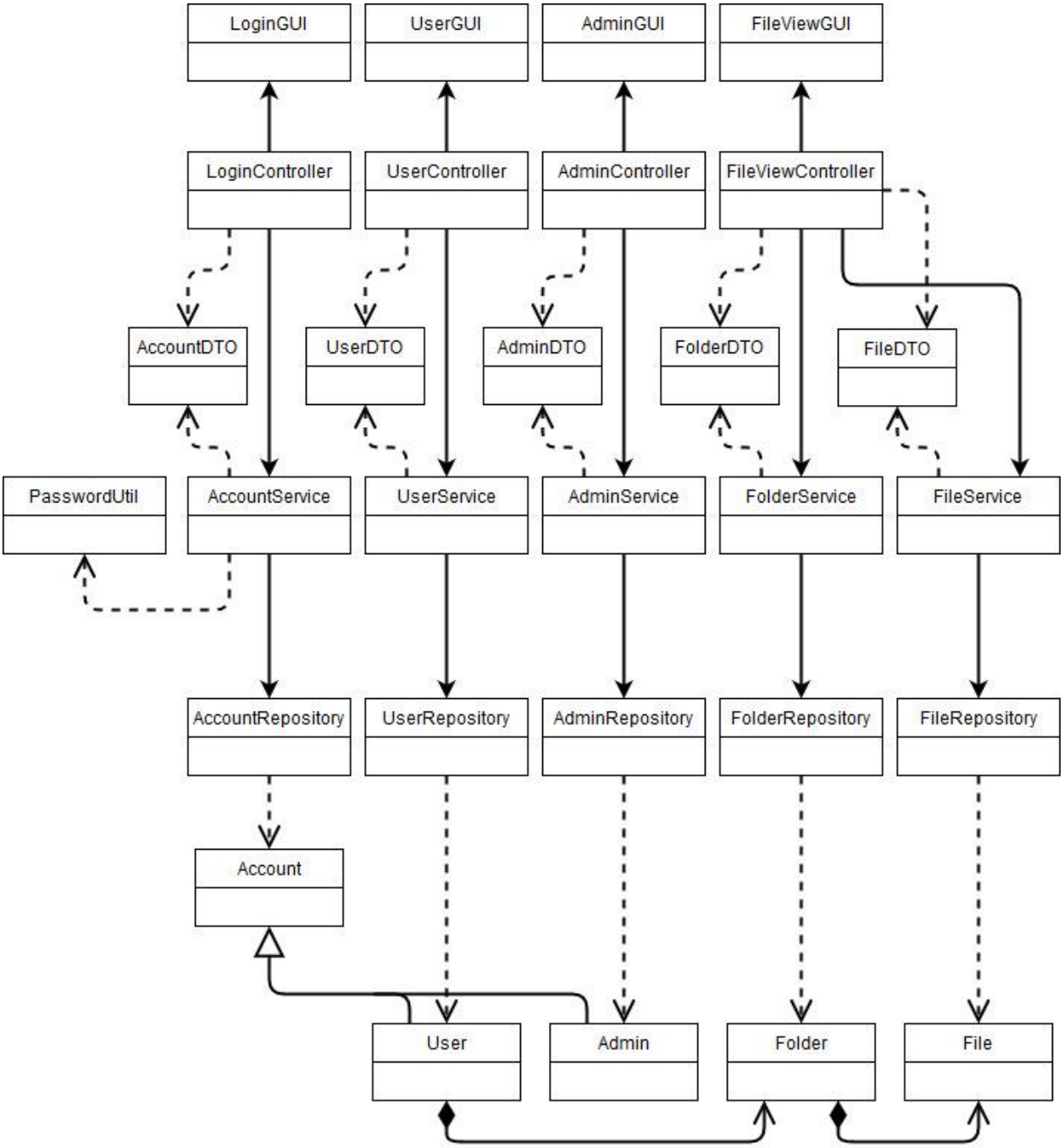
The second scenario represents the user uploading a file into the system:



1.2 Class Design

The following patterns will be applied when implementing the application:

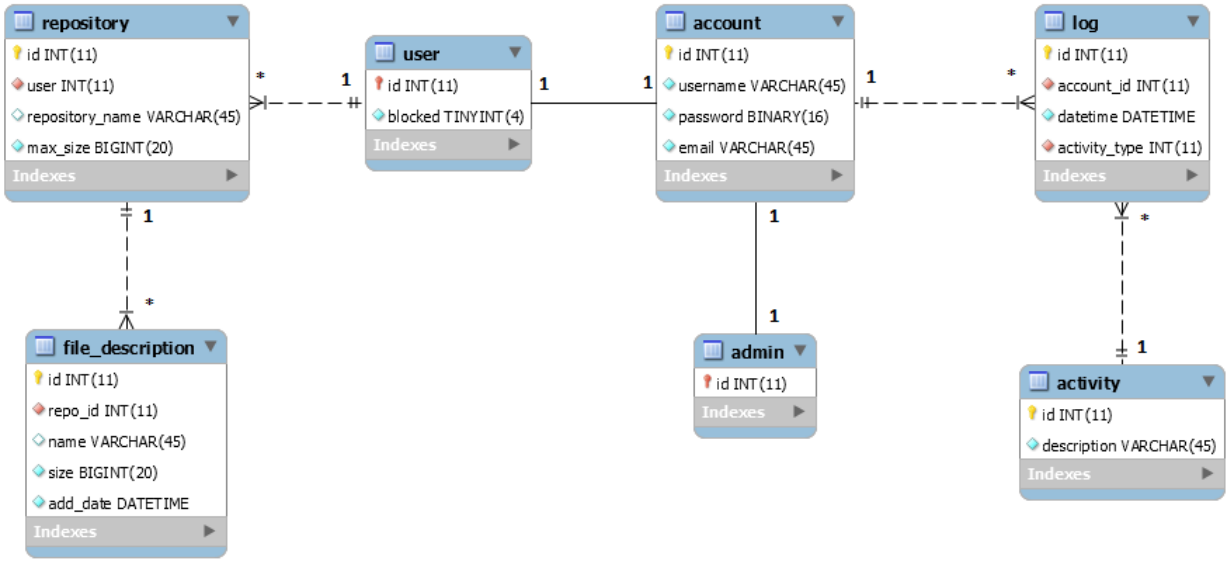
- the Observer pattern in the presentation layer, the GUI classes being the Observable and notifying the controllers of changes
- the Builder pattern may be used for some entities or data transfer objects to facilitate their creation



2. Data Model

The account data, activity logs and file descriptions associated to each user will be stored in a relational DBMS, such as MySQL. The actual files will however be kept in a NoSQL DBMS, such as MongoDB.

The data model for the relational database is the following:



3. Unit Testing

The following methods will be used to test the application:

- Unit testing, which involves testing small individual units of code, such as methods. To this end, two testing frameworks will be used: JUnit along with Mockito. Together, these enable the creation of test double objects in automated unit tests.
- Integration testing, which tests multiple software modules working together. It occurs after unit testing. For this purpose, JUnit may be used, along with an extension such as DbUnit, which allows us to put the database into a known state between test runs.

IV. Elaboration – Iteration 2

1. Architectural Design Refinement

[Refine the architectural design: conceptual architecture, package design (consider package design principles), component and deployment diagrams. Motivate the changes that have been made.]

2. Design Model Refinement

[Refine the UML class diagram by applying class design principles and GRASP; motivate your choices. Deliver the updated class diagrams.]

V. Construction and Transition

1. System Testing

[Describe how you applied integration testing and present the associated test case scenarios.]

2. Future improvements

[Present future improvements for the system]

VI. Bibliography