MU Student Applications Security Request System Requirements Analysis

10-26-2015

"Zou Gotta Do It"

TeamX

University of Missouri - Columbia

CS 4320: Software Engineering

Final Project

Table of Contents

Version History

Version Number	*Implemented By*	*Revision Number*	*Description of Changes*
1.0	*TeamX*	*10/26/2015*	*Initial Version*

2. Glossary

PawPrint: PawPrints are IDs that are given to all of the faculty/employees and students-either employed, enrolled, or both-at the University of Missouri - Columbia. This ID is used to uniquely identify a person at the university.

Requestee: A paid employee of the University of Missouri - Columbia who wishes to file a security request form.

System Admin: A faculty member who has been given the privileges to access log information/files within the university servers.

Head of Department/Dean: The person who is charge of the department which has the right to approve or decline a user's security request forms presented to them.

3.1 Introduction:

3.1 Current System:

The purpose of this software is to simplify the process of gaining security access to records at Mizzou. Gaining this access is currently a complicated and unsafe process. The form has to be filled out by hand, and the person attempting to gain access has to find the department head and/or dean to get relevant signatures before turning the form into Jesse Hall. This was a hassle for anyone who needed to get quick access to academic, admissions, or financial records.

3.2 Proposed System:

Our solution seeks to simplify this process immensely by improving the process in four ways. First, we are looking to improve reliability and readability of the information by replicating the form into a web application. Second, we are attempting to hasten the overall process by allowing for information to be autofilled based on SSO (pawprint) and employee ID. Next, we are attempting to make the process more secure by logging into an account that only allows access to a specific Requestee's previous requests and storing all information into a relational database. Finally, optimizing this process is creating a pathway to making this process more environmentally friendly. When electronic signatures are introduced, this process could potentially become completely paperless. After each request is filled out and submitted, a PDF will be auto-generated and stored into a file system. This file can be retrieved and downloaded after login. The form will still need to be printed out, and signatures will still be required to be retrieved by hand.

3.3 Intended Audience

This application is intended to be utilized by the faculty of the University of Missouri - Columbia.

3.4 Product Scope

This software will affect the faculty of the university such that submitting security forms can be easily generated, printed, and delivered to be approved by the Head of the Department.

3.5 Product Perspective

This product is intended to replace the current system security request system in order to create a more secure, simple, and most ideal software to enable faculty to fluidly generate form releases, print them out, and have the approved, or denied, by the head of the desired department.

3.6 Product Functions

The main function of this product is to allow a Requestee to easily generate, as well as update, a security request form such that it can be printed and delivered to the head of the department for authorization.

3.7 User Classes and Characteristics

The intended users of the software include faculty of the University of Missouri -Columbia-who wish to file a security release form. Along with a system admin for support to assist in managing logs and corrupted files, and provide support where it is needed.

4. User Functional Requirements

4.1 General Requirements

- All users on the system must have a pawprint. This pawprint is needed to log into and use our software.
- The user must also be a faculty member of the University.
- The Software must be used on University-owned servers.

4.2 Requestee Functional Requirements

- The Requestee should have the ability to create a new release form request.
- Not only must they have the ability to create, but they must also have the ability to update a form the have created in the past as well.
- The Requestee must be able to create multiple form requests.
- A Requestee must be able to delete their form requests.
- Finally, a Requestee must be able to submit their new/updated forms.

4.3 System Admin Functional Requirements

- The System Admin should have the ability to access logs of the application use.
- They must be able to have access to form requests to check for potentially corrupted files.

5. User Non-Functional Requirements

5.1 General Requirements:

- Pawprints and Employee ID's must be taken from University Servers (they cannot be created via the application software).

5.2 Requestee Non-Functional Requirements:

- The Requestee must have the pawprint AND employee ID to initially log into the system.
- They should be able to view the creation dates and information that they have filled out previously.
- The Requestee must have the pawprint AND employee ID to initially log into the system.
- They may have the option to change password after the initial login.

5.3 System Non-Functional Requirements:

- The System Admin should have access to all file folders.
- He/She should be able to search through the system/database in a secured manner without having to put the system on hold.

6. System

6.1 System Requirements/Constraints - Client

Our software is able to support cross-platform compatibility. A user should be able to use/access our software from any standard Operating System, such as Windows 7+, Linux, and Mac. The application should be able to run on a desktop, laptop, and also on a mobile device through any of the following browsers: Google Chrome, Mozilla FireFox, Internet Explorer 8 or greater and Microsoft Edge.

6.2 System Requirements/Constraints - Server

Our system is designed to be broadly based on client server architecture. We will utilize php and postgresql to carry out transactions. PostGreSql database is used to store login information, as well as new & pre-existing requests from a Requestee of which will be generated into a pre-defined PDF. Because of this, a PostGreSQL server will be needed to correctly deploy our software. The ability to host our app on a server repository will also be necessary, hence the needed php execution. A filesystem will also be needed in order to store all pre-generated PDFs made via request by the Requestee.

6.3 System Requirements/Constraints - Hardware

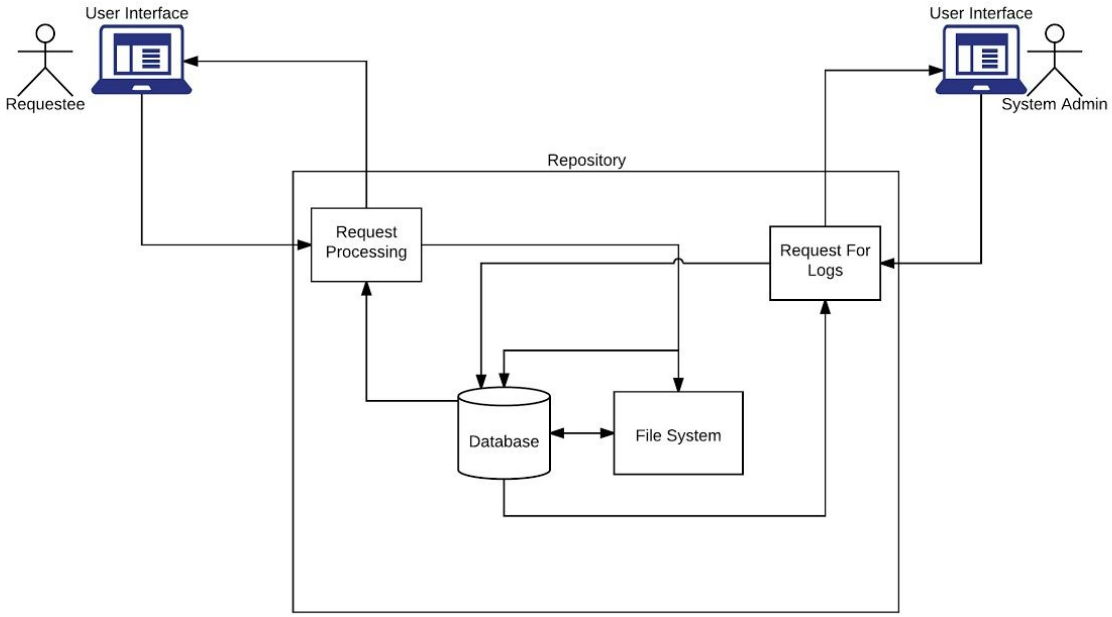
To use this software, client-wise, hardware specifications are very minimal. Server-wise, the host must have repositories and databases capable of supporting our system; as well as a network for data to be transferred over. Size/processing speed may vary based on expected number of transactions taking place.

6.4 Security Requirements

To ensure best security, transactions will be done via a HTTPS protocol. It is recommended that the host have a System Admin to manage and log server transactions. This will prove handy if proof is needed to resolve any foul play. To provide an additional layer to security, a user has a maximum of 4 attempts to correctly enter their username and password before the account goes on hold and needs to request for an unlock on the account

7. Architecture Diagram

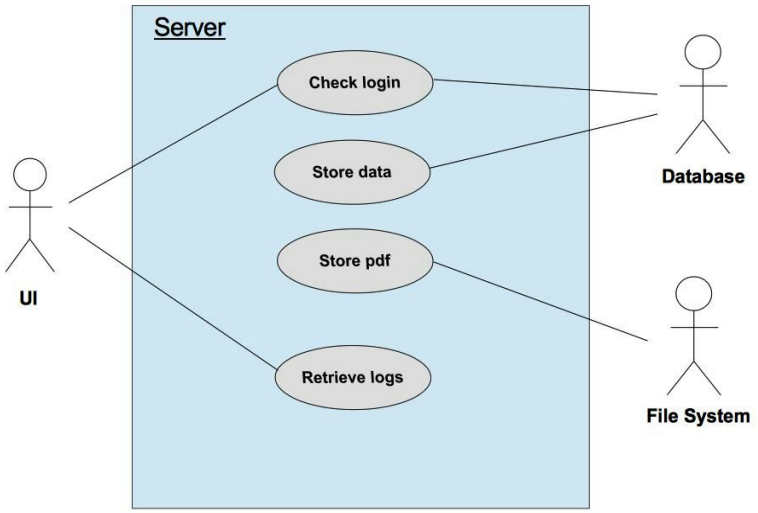
This diagram display the structure of the software. There will be a UI for both the...



Requestee and the System Admin. The internal workings includes a repository of which has a Database and file system. Diagram also displays the flow of data between these components.

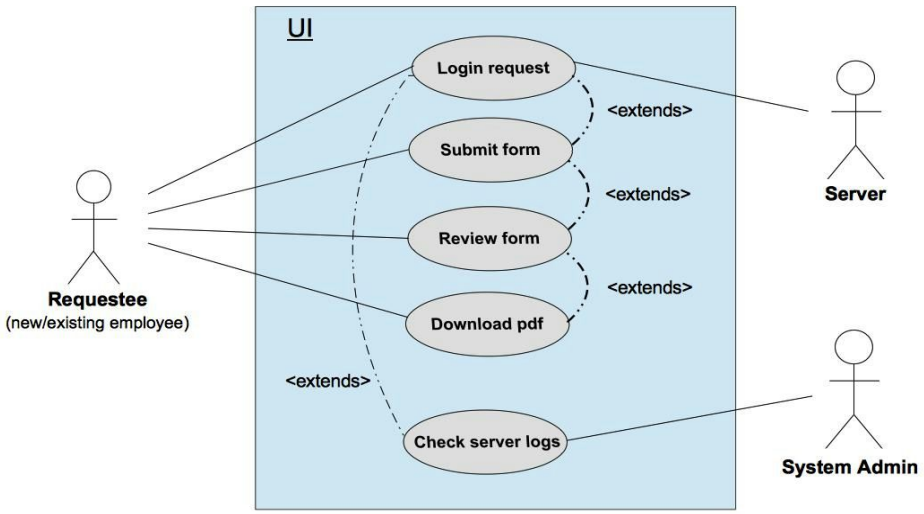
8. Use Case Diagrams

8.1 Use-Case(Server):



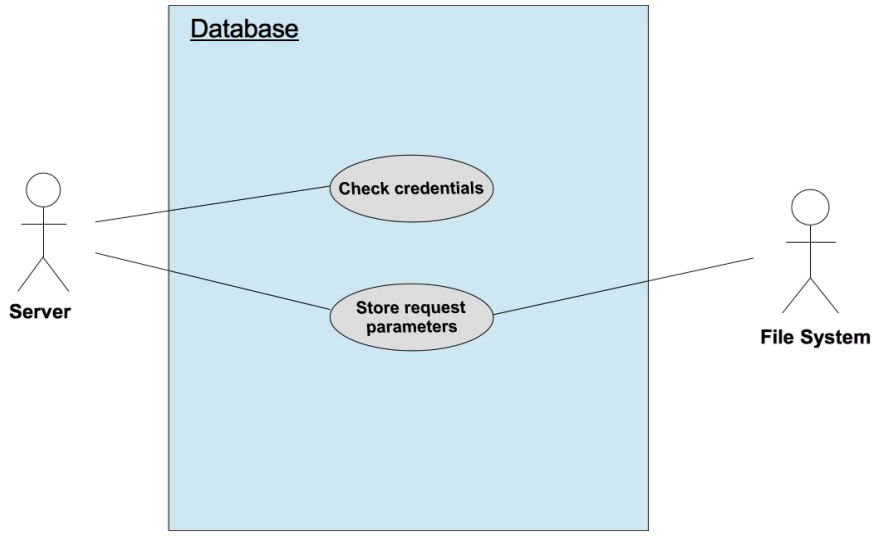
This diagram shows the actors (UI, Database, and System Admin) and their interactions with the server

8.2 Use-Case(User-Interface):



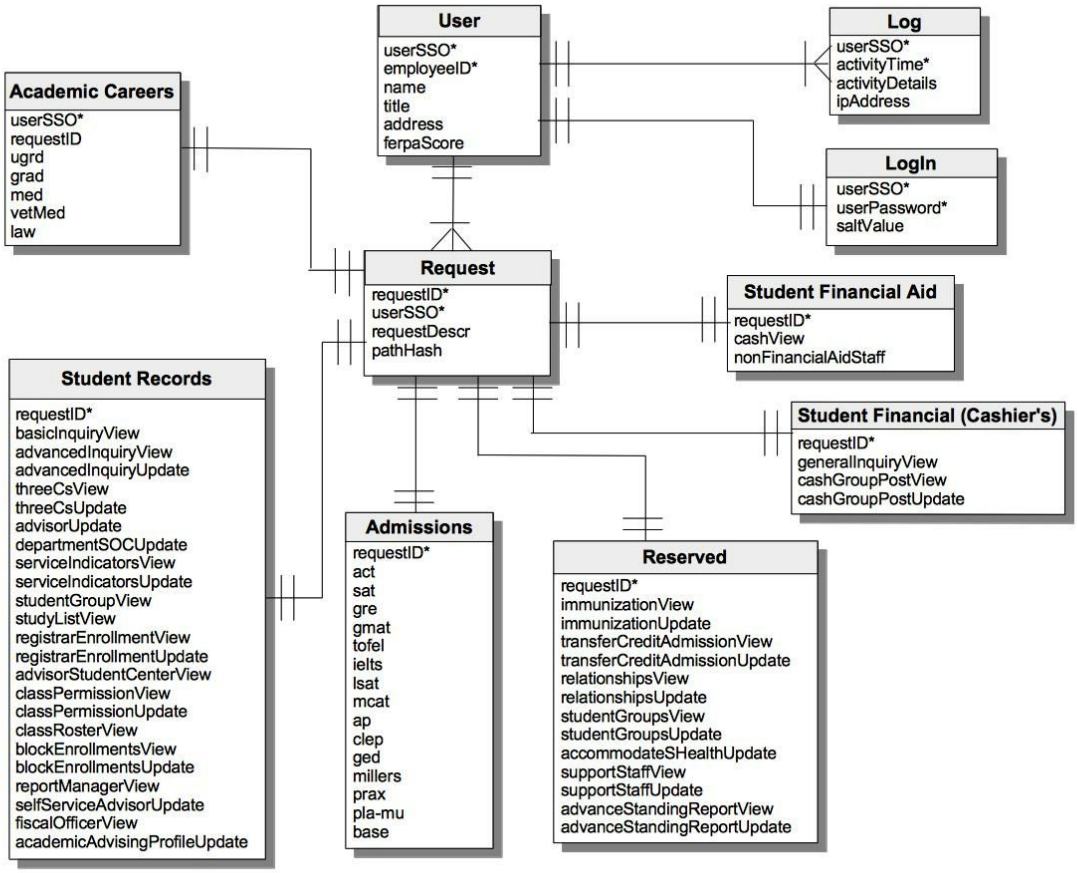
This diagram shows the actors (Requestee, Server, System Admin) and their interactions with the user interface.

8.3 Use-Case(Database):



This diagram shows the actor (Server and File system) and their interactions with the Database.

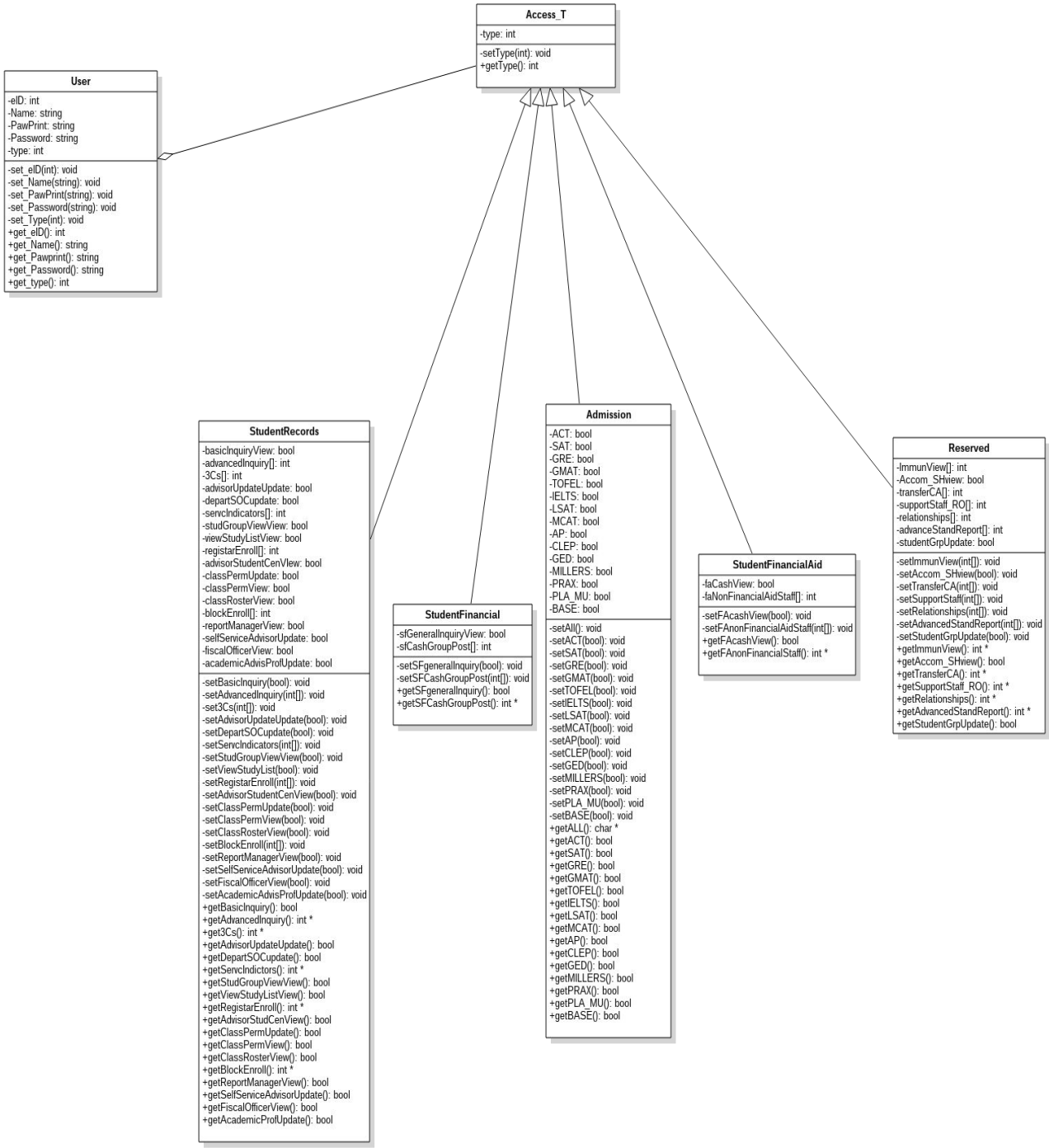
9. ERD Diagram



This diagram represents the structure and organization of the database for the software, using crow's foot notation to represent cardinality and relationships between tables.

10. Class Diagram

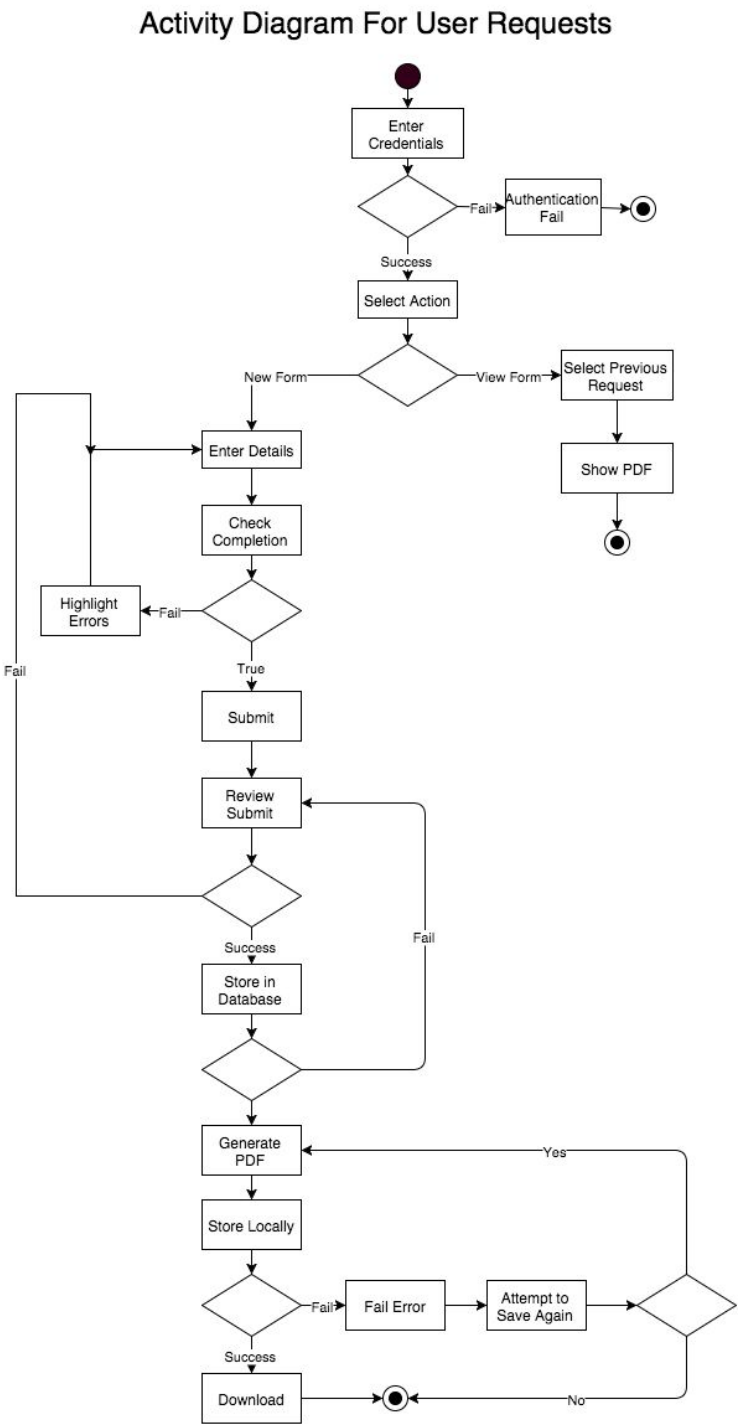
This diagram outlines the objects used in the web-based applications. User is defined as such: both requestee and System Admin. The Access_t is a super-class of the 5 sub-classes of the 5 different applications for security request form



11. Activity Diagrams

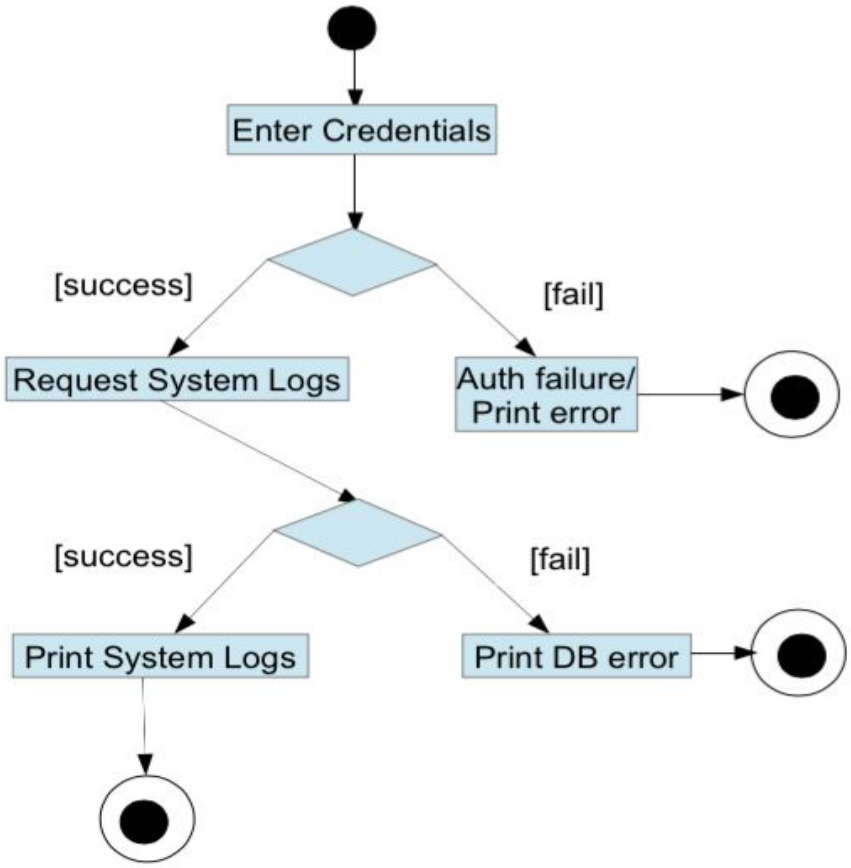
11.1 Activity Diagram (Requestee):

The following diagram outlines the flow of decisions by the requestee.



11. 2 Activity diagram(System Admin)

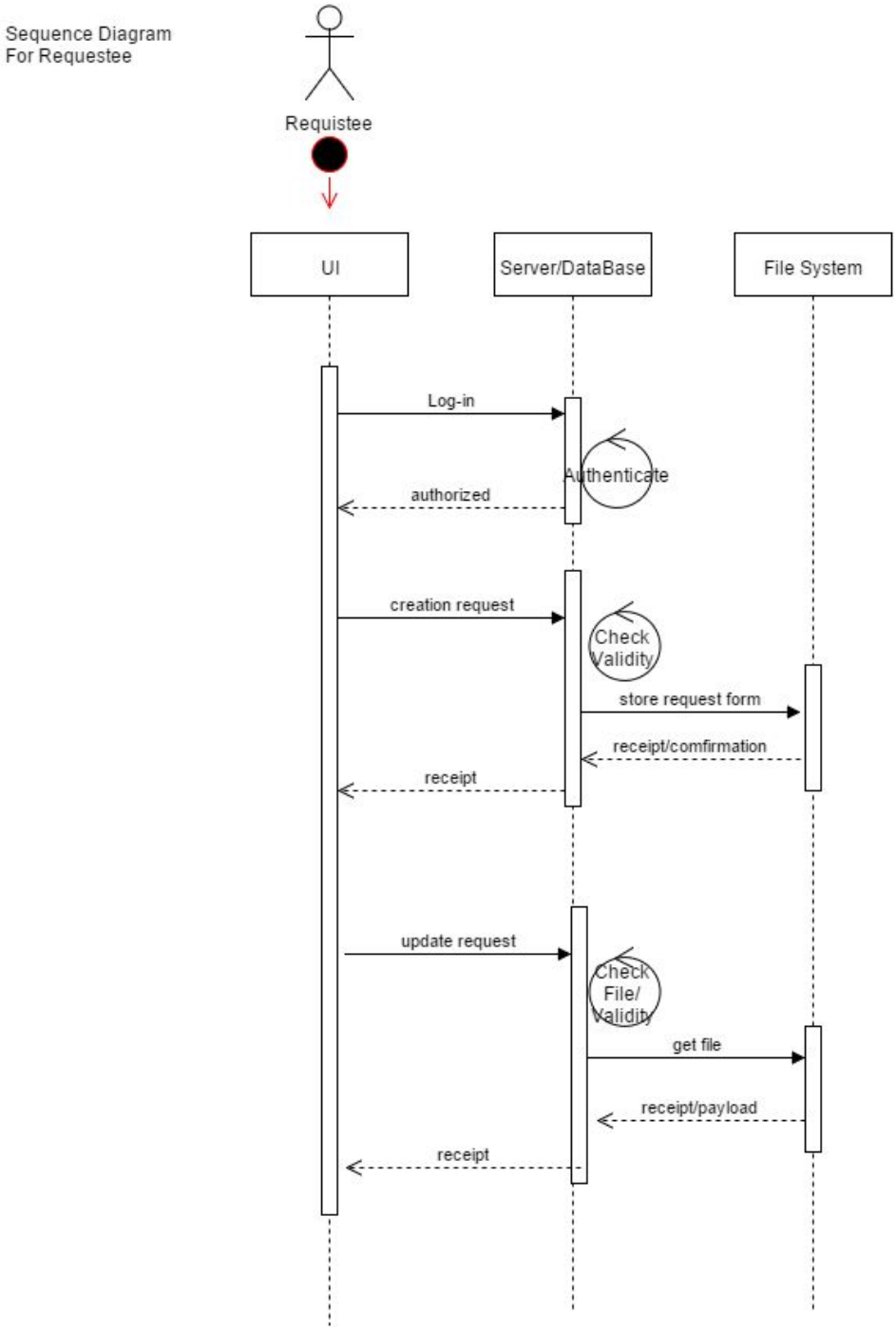
This diagram shows the flow of decisions by the system admin.



12. Sequence Diagrams

12.1 Sequence Diagram - Requestee

The following diagram represents how a Requestee's session is implemented. As a requestee uses the interface, it communicates/interacts with the Server/Database and FileSystem. Note the verification processes steps take in order to ensure security and validity of particular actions.



12.2 Sequence Diagram - System Admin

The following diagram represents how a System Admin's session is implemented. As a System Admin uses the interface, it communicates/interacts with the Server/Database and FileSystem. Note the verification processes steps take in order to ensure security and validity of particular actions.