ESOF 322: Project 1 - Cold Case Database

Jacob Coleman, William Jardee, Fletcher Philips, Megan Steinmasel

December 1, 2022

System Description: In the world of criminal justice, it is not a rare occurrence that the truth is thinly veiled by a lack of gathered evidence or proper techniques to analyze current data. A prime example of this is the aid of DNA testing in prosecution. The proposed software system will collect old and new evidence into one database (a centralized storage system), organize it, and present it to domain specialists through an easy-to-use website. The initial construction of this project is humble but aims to create a skeleton for more sophisticated analysis techniques to be built on. It is assumed that there are two primary ways to interact with the system: through control of the database operations and in the form of the website; these are the database engineer and website user, respectively.

Database Engineer: An individual knowledgeable of how the system works and responsible for maintenance and updates.

Website User: A front-end user that likely lacks data science knowledge traversing the web system.

Criminal Justice Professional: Someone in the field of criminal justice who will be using our system to make their job easier (This is synonymous with Website User).

User Stories

Epic: As a Criminal Justice Professional, I want a database that can hold, sort, and present data according to a variety of different queries and eventually use state-of-the-art algorithms to help me succeed in my job.

User Story 1 (Megan Steinmasel): As a website User, I want a search bar that will search the database.

User Story 2 (William Jardee): As a website user, I need the processing of data to be done automatically in a way that is friendly to someone that is not a data scientist.

User Story 3 (Jacob Coleman): As a website user, I want a data visualization technique that is intuitive and accurately shows patterns in the data.

User Story 4 (Fletcher Philips): As a database engineer, I want to have a central storage system so that I can store cold cases inside of it.

User Story 5 (Fletcher Philips): As a database engineer, I want a central storage system so that the admin can insert and delete data.

UseCase Diagram

Author: William Jardee

It should be noted that while William created this iteration of a UseCase Diagram, it is the combination of numerous individual diagrams made by each group member.

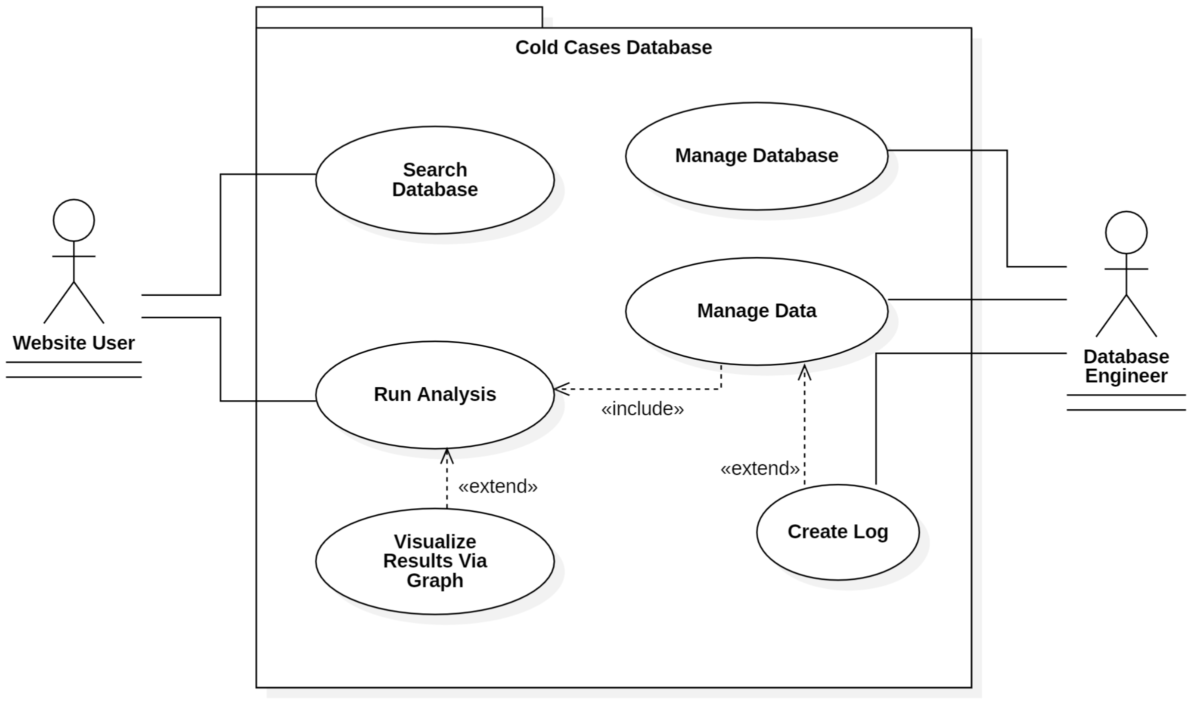

Figure 1: UseCase Diagram for the Cold-Cases Database system.

Textual Descriptions

Author: Megan Steinmasel

Name:	Search Database
Description:	Search bar scans keywords through the database so the Website User can find a desired query/page.
Related Requirements:	User Story 1
Preconditions:	The user has successfully logged onto the website.
Successful end condition:	Successful display of search bar and successful redirection.
Failed end condition:	Search bar is not displayed or failed redirection.
Actors:	Website User
Basic Flow of Events:	1. Actor types in keywords to the search bar and clicks enter or selects a page to navigate to. 2. Request is processed, and the database is searched. 3. The appropriate query/page is displayed.
Extensions/Exceptional Flow of Events:	1. An error is recognized: the search bar is not displayed and/or failed redirection. 2. Actor is notified of an error. 3. Log data. 4. The Actor is prompted for a refresh of the webpage. 5. When the refresh button is pressed, the system reboots and tries again.

Author: Fletcher Philips

Name:	Manage Database
Description:	The database that contains all the data will be maintained. This included initial creation, implementation of new features, and regular maintenance.
Related Requirements:	User Story 4
Preconditions:	Basic web infrastructure has been created. A small sample set of cold cases is ready to be inserted. The current actor has the proper credentials.
Successful end condition:	The database can be manipulated in the desired manner.
Failed end condition:	Requested change cannot be done.
Actors:	Database Engineer
Basic Flow of Events:	1. The current state of the database is displayed. 2. A new change or maintenance is done to the system and attempted to be launched. 3. The new state of the system is valid and becomes live.
Extensions/Exceptional Flow of Events:	1. Database fails to build properly. 2. The engineer is notified that an error has appeared. 3. Log data. 4. Engineer is returned to the state before the change was added.

Author: William Jardee

Name:	Manage Data
Description:	Data needs to be inserted, deleted, and manipulated. Related to this, there must be an appropriate interface to do this through. The data will connect directly with the database.
Related Requirements:	User Story 2, User Story 5
Preconditions:	The user is logged on and has gained access rights according to their credentials.
Successful end condition:	Data is successfully manipulated, and a success message is received from the source.
Failed end condition:	Requested task is outside of credentials. Success message not received. Invalid new data.
Actors:	Database Engineer
Basic Flow of Events:	1. Actor selects the action they wish to do and what to do it on. 2. Change is enacted. 3. Flow is complete and prompts the user for the next action.
Extensions/Exceptional Flow of Events:	1. Conflict happens. 2. Reject any attempted changes and revert any changes. 3. Notify the actor that there has been an error and log data. 4. Flow is complete and prompts the user for the next action.

Author: William Jardee

Name:	Run Analysis
Description:	An understandable interface for viewing data and selecting data visualization.
Related Requirements:	User Story 2, User Story 3
Preconditions:	The website user has signed into the website and accessed the drop-down menu to access the page.
Successful end condition:	A valid group of data is selected, a visualization format is selected, and the servers are available for the request.
Failed end condition:	One of the above three requirements has been violated, and the window is closed.
Actors:	Website User
Basic Flow of Events:	1. A query selection screen is presented, and the desired data is collected. 2. The type of visualization is selected from a dropdown menu. 3. The request is sent to the "Visualize Results via Graph" action.
Extensions/Exceptional Flow of Events:	1. One of the system-side error states was reached. 2. Report the error to the actor. 3. Log data. 4. Return actor back to the homepage.

Author: Jacob Coleman

Name:	Visualize Results via Graph
Description:	Displays selected graphs from the gathered data
Related Requirements:	User Story 3
Preconditions:	The website user has signed into the website and accessed the page. Some data has been selected for analysis, and the desired type of graph has been selected.
Successful end condition:	The graphs are displayed on a separate page to view.
Failed end condition:	Fails to display any graphs.
Actors:	Website User
Basic Flow of Events:	1. The website user goes through a data selection process. 2. The website user goes through a visualization selection menu. 3. The system will display any applicable graphics.
Extensions/Exceptional Flow of Events:	1. The graphics fail to display. 2. The User is notified that the user cannot access graphics. 3. Log data. 4. User is sent back to the homepage.

Author: William Jardee

Name:	Create Log
Description:	A log file should be kept to track flow and diagnose errors and suspicious behavior.
Related Requirements:	Catch all location for all errors (no specific user story)
Preconditions:	The system has been started effectively, and there is a safe place to store a text file (log file).
Successful end condition:	Data can be saved to the log file.
Failed end condition:	Data cannot be safely saved to a log file.
Actors:	Database Engineer
Basic Flow of Events:	1. Write to file recent activity. 2. Flag any invalid actions prompt “Log data.”
Extensions/Exceptional Flow of Events:	1. Notify the system admin of the issue and include error information. 2. Terminate all systems until the issue is resolved.

Class Diagram

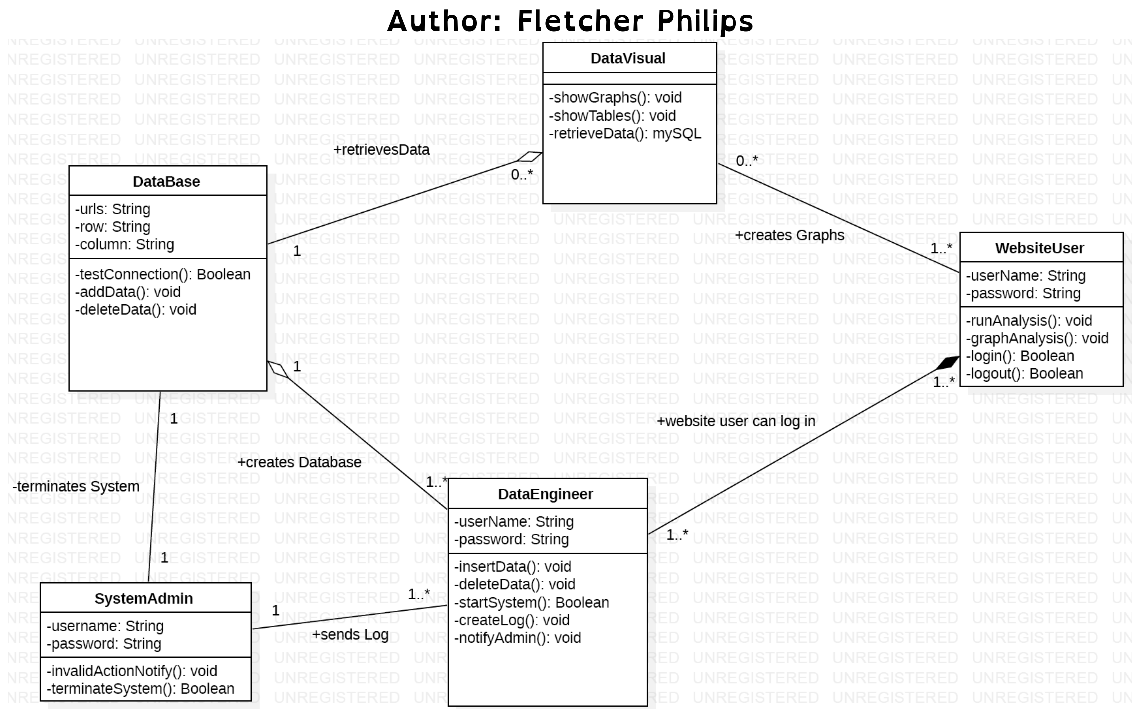

Figure 2: Class Diagram for the Cold-Cases Database system.

State Chart Diagram

Author: Megan Steinmasel

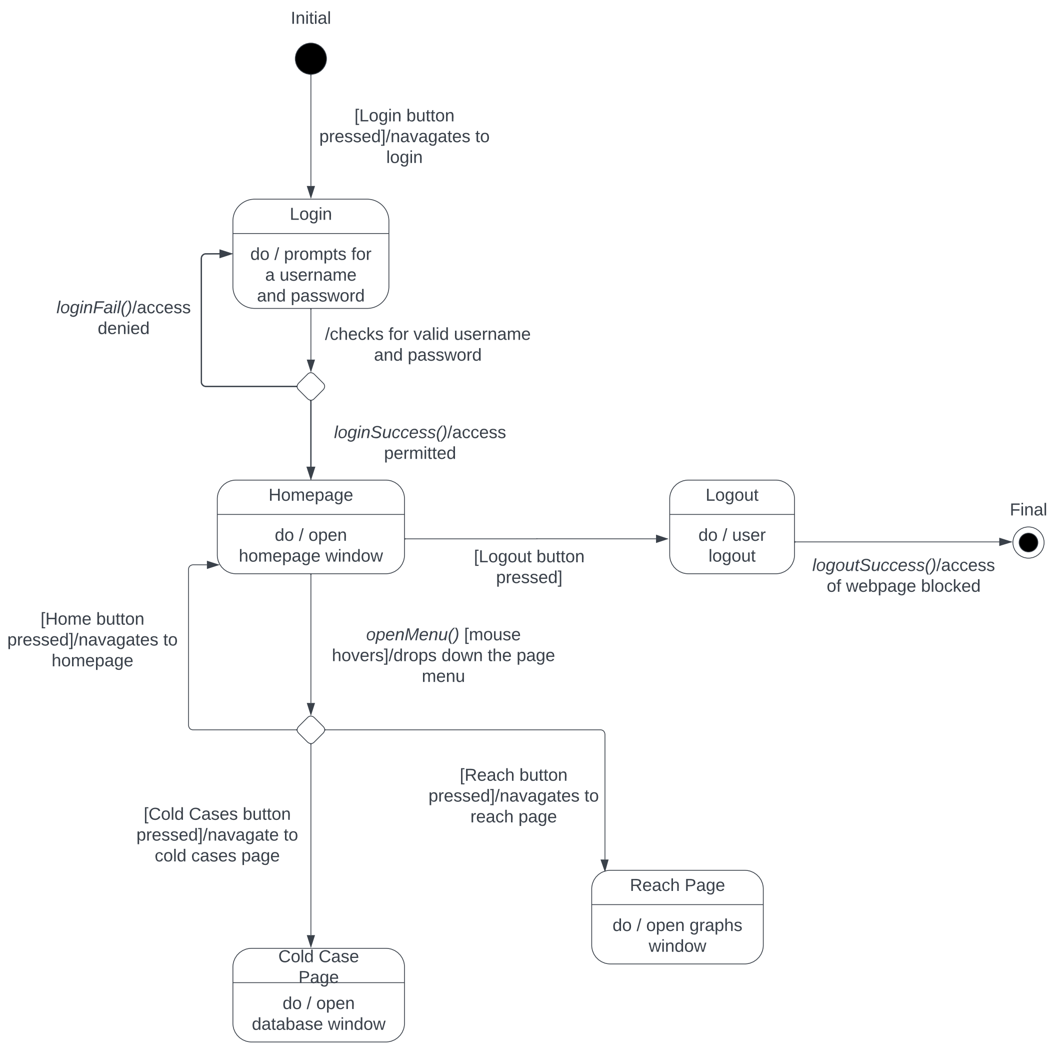
Initial
[Login button pressed]/navagates to login
Login
do / prompts for a username and password
loginFail()/access denied
/checks for valid username and password
loginSuccess()/access permitted
Homepage
do / open homepage window
[Logout button pressed]
Logout
do / user logout
logoutSuccess()/access of webpage blocked
Final
[Home button pressed]/navagates to homepage
openMenu() [mouse hovers]/drops down the page menu
[Reach button pressed]/navagates to reach page
[Cold Cases button pressed]/navagate to cold cases page
Reach Page
do / open graphs window
Cold Case Page
do / open database window

Figure 3: State Chart Diagram for the Cold-Cases Database system.

Activity Diagram

Author: Jacob Coleman

Database Engineer

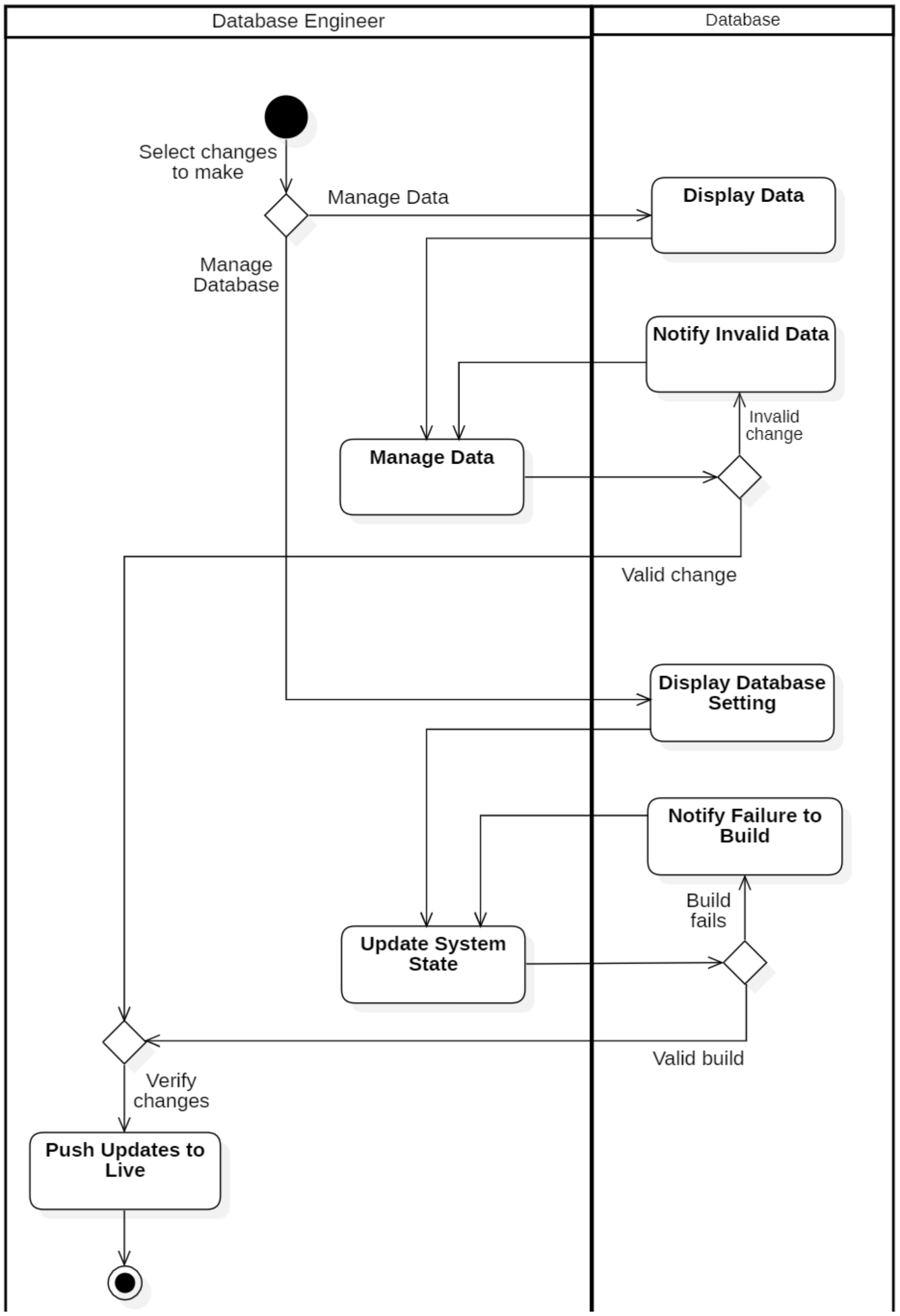

Figure 4: Activity Diagram for the Database Engineer workflow in the Cold-Cases Database system.

Author: Jacob Coleman

Website User

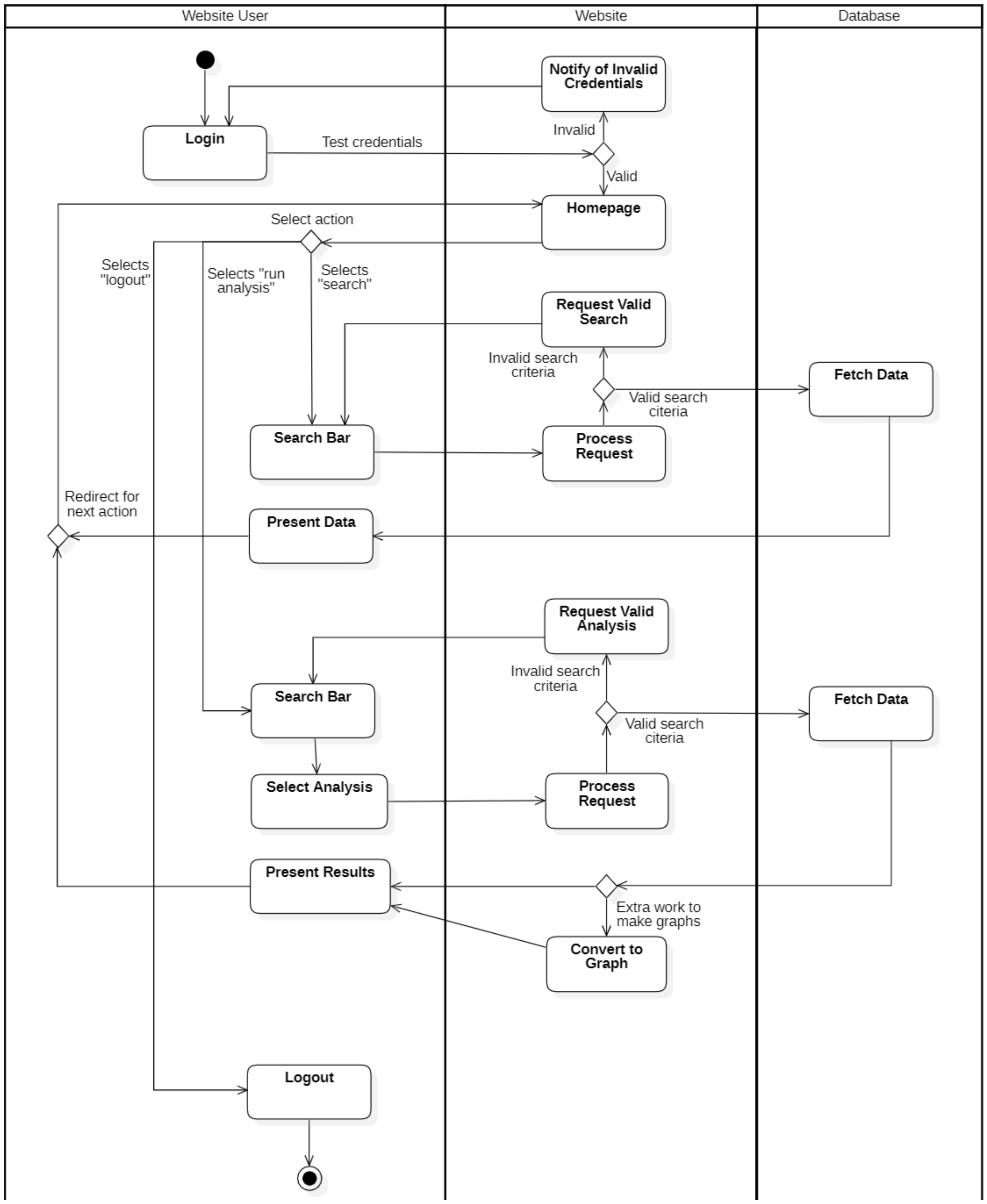

Figure 5: Activity Diagram for the Website User workflow in the Cold-Cases Database system.