Team 6's Semester Project: Fishing Database

CS 4300 – Software Design
Semester: Fall
Final Presentation
Ben Wilmer, Gavin Stierstorfer

Professor: Dr. Armin Moin

Project Description

- **Fishing Database**
- **Plan-driven Path**
- **Allows users to store catch information**
- **Includes: Location, lure, type of fish, pictures, size, etc… (Not All Info needed)**

User Stories

- **Problem: No way to easily store caught fish information**
- **Users: Casual Fishermen, Pro Fishermen**
- **Pro Fishermen can use it as an analytical Tool**
- **Could be expanded for general wildlife**
- **Information from database can be used to understand patterns in fishing**

Related Works/USP

- **Other databases such as the IGFA Database store general fish information**
- **Other apps such as Catchr and FishVerify store limited information about the fish**
- **Our database lets you store your own catch information**

Iteration Schedule

Weeks	Task - Gavin	Task - Ben	Completed?
9/29/25 - 10/5/25	Develop Full Features List	Develop Full Features List	Yes
10/6/25 - 10/12/25	Create Pseudo Code	Decide on Task List	Yes
10/13/25 - 10/19/25	Open-Source Decision	Open-Source Decision	Yes
10/20/25 - 10/26/25	Meeting to see progress	Meeting to see progress	Yes
10/27/25 - 11/2/25	Complete Diagrams	ADD Iterations	Yes
11/3/25 - 11/10/25	Improve Diagrams	ADD Iterations	Yes
11/11/25 - 11/17/25	Fill Power Point	Fill Power Point	Yes

Task List

- **Create detailed UML diagrams**
- **Fully understand wanted features and logic**
- **Create light pseudocode**
- **Decide on open source**
- **Schedule meetings**
- **Due to doing a plan-driven path, most of our task and works will be a combined effort**

Completed Task

- **Completed UML Diagrams**
- **Full features list**
- **Light Pseudocode**
- **Open source**
- **ADD Iterations**

Full Feature List

- **User Login and information storage**
- **Main Dashboard**
- **Add Fish (includes other information)**
- **Private/Public options**
- **View Catalog and private**
- **Edit Catalog**
- **Search for fish or locations**

Pseudocode

User logs in or creates an account
User enters the main dashboard
If the user selects Add Fish:
User enters fish details (species, location, etc.)
User chooses public or private
Fish is stored with visibility setting
If the user has more fish to add:
Repeat the add process
Else:
Return to dashboard
If the user selects View Catalog:
User sees all cataloged fish
User may view both personal and public entries
If the user chooses to edit a fish entry:
User updates fish information
Changes are saved
Else:
Return to dashboard
If the user selects Search:
User searches by species or location
Matching fish entries are shown
User may view details or return to dashboard
If the user selects Logout:
Session ends
Application stops

Open-Source Licensing (MIT)

- Copyright 2025 Ben Wilmer, Gavin Stierstorfer
- Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the “Software”), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:
- The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.
- THE SOFTWARE IS PROVIDED “AS IS”, WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

GitHub Repository

- **Gstierst/CS4320-project - Link to GitHub Repository**
- **Where we stored documentation and diagrams**

Project to Course Correlation

- **Our project correlates to the course by documenting the design of an online database using multiple graphical designs that were provided using the course's content**

ADD 3.0

- **2 Iterations**
- **Iteration 1 focuses on the core database functionality**
- **Iteration 2 refines the functionality and adds data caching and sharing**

ADD Use Cases

Use Case	Description
UC-1: Store catch information	Casual and Professional fishermen can store the data of fish they cache in the database. They can choose whether or not to share this data with other users on the platform.
UC-2: Extrapolate and analyze fish information	Professional fishermen can utilize their catch data to determine when the best time to fish is, the best locations to fish at, and the best bait to use for a specific type of fish.
UC-3: General wildlife information storage	All users can utilize the platform to document wildlife sightings other than fish, such as a sighting of a rare/near-extinct animal.
UC-4: Sharing caches with other users	Users can elect to share their catch information with other users on the platform on a case by case basis.

ADD Quality Attributes

ID	Quality Attribute	Scenario	Associated Use Case
QA-1	Security/Privacy	The user should be able to select whether or not to share their catches with other users. This preference can be edited at any time.	UC 4
QA-2	Security/Privacy	The user's entered personal information should be protected and hidden at all times.	All
QA-3	Availability	The user should be able to access their fishing information at any time and should have offline access if necessary.	UC 1-3
QA-4	Performance	Users should be able to perform normal operations on the database quickly, even when many other users are connected.	All
QA-5	Performance	The database should consistently update with new information every 5 minutes.	UC 1-3
QA-6	Scalability	The database will store catch information for extended periods of time. Allows for viewing of previous year's catches for data extrapolation.	All
QA-7	Deployability	The system will be deployed into production at all times, with separate environments for development and testing to ensure that the service is always running.	All

ADD Constraints

ID	Constraint
CON-1	Large amounts of data must be stored in the database in the event that many users register for the service.
CON-2	The database that stores the fishing information is online only, so it cannot be accessed without an internet connection.
CON-3	Information that has been chosen to be shared must be stored separately from private information.

ADD Concerns

ID	Concern
CRN-1	Getting the database up and running, along with allowing CRUD operations for multiple users.
CRN-2	Potential for offline storage/access of fishing data.
CRN-3	Entering fishing information into the database manually, does not lend well to expansion.

ADD Iteration 1 Step 1

Prioritize Quality Attributes, Primary UCs: UC 1 and 3

Scenario ID	Importance to the Customer	Difficulty of Implementation According to the Architect
QA-1	High	Medium
QA-2	High	Medium
QA-3	Medium	High
QA-4	Medium	Medium
QA-5	High	Low
QA-6	High	Low
QA-7	High	Medium

ADD Iteration 1 Step 2

Establish iteration goal by selecting drivers

- **Goal: Basic database functionality and overall system structure**
- **CRN-1**
  - **CONs 1-3**
  - **QA 1-2: Security/Privacy**
  - **QA 3: Availability**
  - **QA 4-5: Performance**
  - **QA 6: Scalability**
  - **QA 7: Deployability**

ADD Iteration 1 Step 3

Choose one or more elements of the system to refine

- **Initially the entire system needs to be defined**

ADD Iteration 1 Step 4

Choose one or more design concepts that satisfy the selected drivers

- **Initially planned on a centralized system design**
- **After consideration, decided that a distributed system design would be better**

ADD Iteration 1 Step 4 (2)

- **Considered the following distributed architectural patterns:**
  - **Leader-Follower**
  - **Multi-Tier Client-Server**
  - **Distributed Components**
  - **Peer-to-Peer**

  Decided on Two-Tier Client-Server (Fat)

ADD Iteration 1 Step 5

Select design decisions and determine rationale

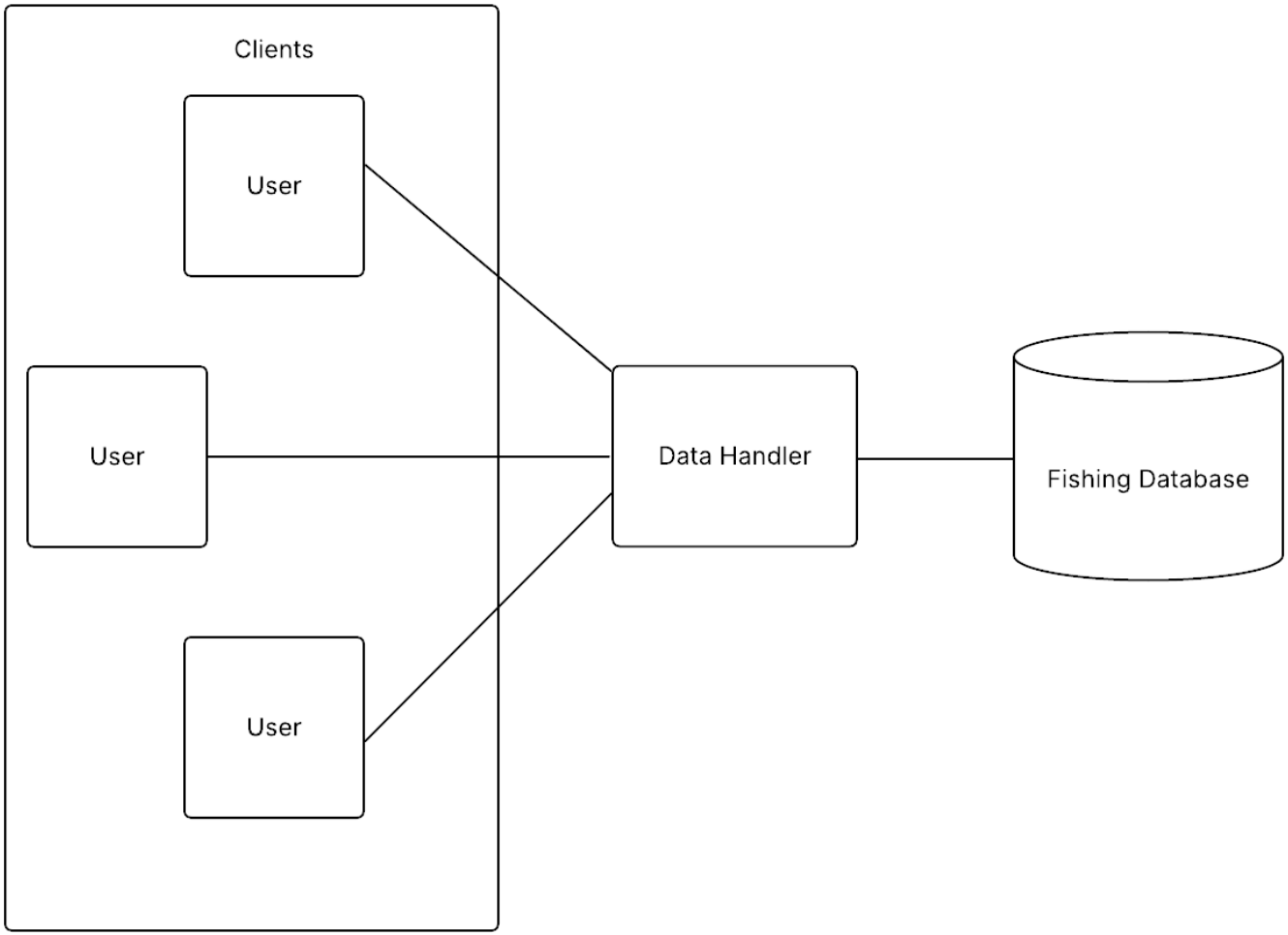
Design Decision and Location	Rationale
Convert to a distributed system by utilizing the Two-Tier Client-Server architecture due to the long-term benefits	The Fat Two-Tier Client-Server Architectural pattern makes use of the available processing power on the device running the software, which allows for additional functionality that can be implemented in future iterations, such as data caching. This conversion was done due to the decision of offline access being undetermined. By converting to a distributed system, this would keep the option open in the future given that it would be a large boon to users.
The central database will act as the transaction server for information, and user devices will act as clients	In line with the Fat Two-Tier Client-Server Architecture, user devices will act as clients which will connect to the central database to receive or send information. This is associated with the following drivers: •Database Storage (UC-1, UC-2, UC-3, QA-5, QA-6, QA-7, CON-1) •Client View with CRUD Operations (QA-2, QA-4, CON-3)

ADD Iteration 1 Step 6

Sketch views and record design decisions

Element	Responsibility
Textual Data Storage	Textual information entered manually by the user. Can be for fishing catches along with other wildlife entries. Fishing data can be analyzed. This data will be stored in JSON format. (UC 1-3, QA-2, QA-6, CON-1, CON-3)
Image Storage	Images of popular formats such as .png can be uploaded and stored on a per-entry basis. This element is part of the Catch View. (UC1 and 3, QA-6, CON-1, CON-3)
Catch View	This view will show the user all of the information relating to the selected database entry. From here, the entry can be edited as desired. (UC 1-3, QA-4, CON-1)
Catch CRUD Operations	Create, Read, Update, and Delete operations. Users will be able to create new database entries, view their previously created database entries, update their previously created database entries, and delete their previously created database entries. Read, Update, and Delete operations are provided through the Catch View. (UC 1-3, QA-4, CON-1)
Drop-Down Menu Views	This element will comprise preprocessed drop-down menus which have the most common and popular fish types, locations, and bait types in the area. This allows for quick selection of attributes for data entries rather than manual entry. (UC 1-2)

Two-Tier Client Server (Fat) Diagram



Iteration 1 Step 7

Perform analysis of current design and review iteration goal and achievement of design purpose

Not Addressed	Partially Addressed	Completely Addressed	Design Decisions Made During the Iteration
	UC-1		Used the Fat Two-Tier Client-Server Architecture to allow users to connect to the database and store/view catch information. No offline access currently.
	UC-2		Textual Data Storage allows pro-fishermen to view important catch information to analyze and extrapolate data to better make decisions. No offline access currently.
	UC-3		Textual and Image Storage allows more general wildlife entries to be made. Only manual entry is available for this feature. No offline access currently.
UC-4			No relevant decisions made.
QA-1			No relevant decisions made.
		QA-2	Any personal information entered by the user will be stored securely on the server. Security features mentioned previously will keep information safe.
	QA-3		Although no offline access exists at this point, caching can be implemented in relation to the selected architecture (TTCS).
		QA-4	Due to the selected architecture, processing load is dependent on the client and not the server, therefore processing load should be well distributed geographically.

Iteration 1 Step 7 (2)

		QA-5	Due to the selected architecture, the database itself should not have much processing load relegated to it, therefore it will be speedy in its updates.
		QA-6	Text and Image Storage allows for long term data storage in the database.
	QA-7		Once the service enters production, a separate branch will be utilized for the implementation of new features.
	CON-1		Since the basic framework for the service has been created, users can store data privately.
CON-2			No relevant decisions made.
CON-3			No relevant decisions made.
		CRN-1	Users can create, read, update, and delete their own database entries.
CRN-2			No relevant decisions made.
	CRN-3		The adoption of the Fat Two-Tier Client-Server architecture means that new features and the expansion of current features can be implemented easier.

Iteration 2 Step 2

- **Goal: Implementation of Key Features**
- **CRN-2**
- **Selected Drivers:**
  - **CONs 2-3**
  - **QA 1-2: Security/Privacy**
  - **QA 3: Availability**

Iteration 2 Step 3

- **The selected drivers, primarily pertaining to the implementation of the data caching and cache entry sharing features will be refined.**

Iteration 2 Step 4

- **Considered the following architectural designs:**
  - **Model-View Controller (MVC)**
  - **Layered**
  - **Repository**
  - **Pipe and Filter**

  Decided on Layered Architecture

Iteration 2 Step 4 (2)

- **Decided on additional design patterns:**
  - **Creational Design**
    - **Prototype**
  - **Structural Design**
    - **Flyweight**
  - **Behavioral Design**
    - **Template Method**
  - **Concurrency Design**

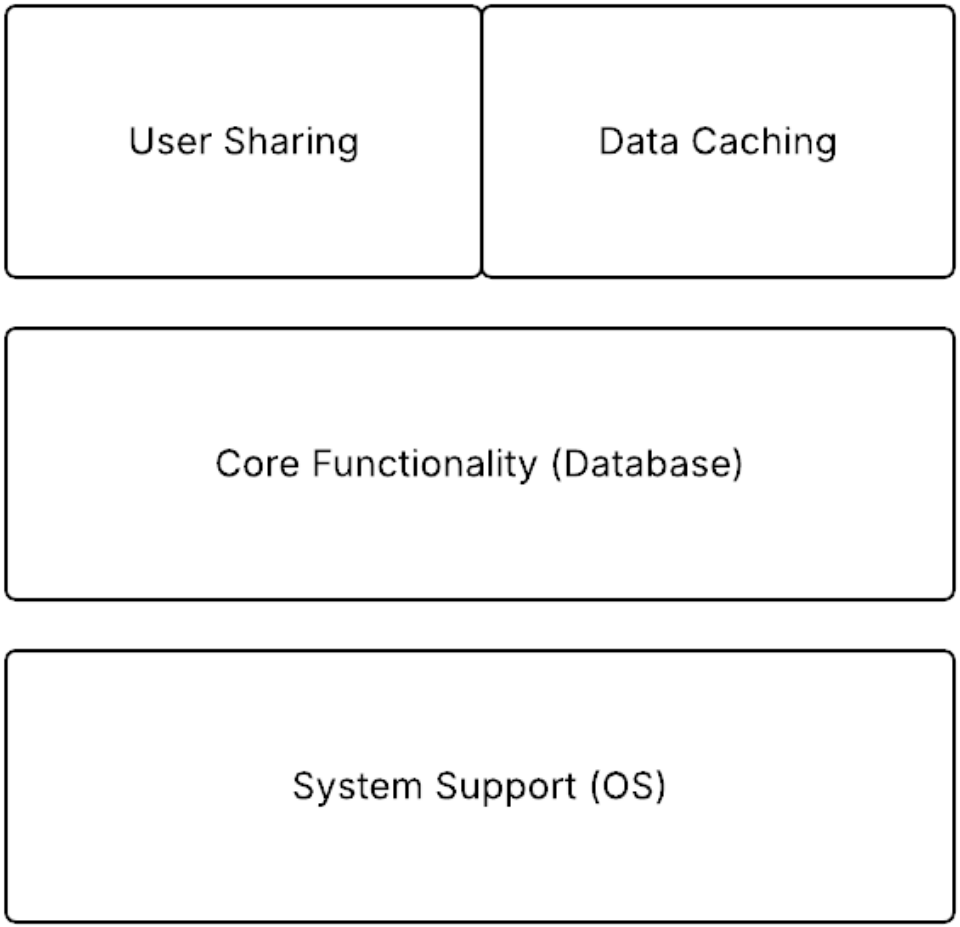
Iteration 2 Step 5

Design Decision and Location	Rationale
Implement data caching in the database	In line with the Layered Architectural design, data caching can be implemented on top of the core features to allow for offline access when an internet connection is unavailable.
Implement catch sharing between users	In line with the Layered Architectural design, catch sharing can be implemented on top of the core features to allow users to make their catches private or public. Public catches can be viewed by other users who are utilizing the service.
Adopt more architectural designs that fit the project	Additional architectural designs that fit the project will be adopted. Namely, the Prototype Creational Design Pattern, Flyweight Structural Design Pattern, and the Template Behavioral Design Pattern will be adopted. These design patterns fit within the design purpose of the project and will help to better focus the scope and development efforts of the project.

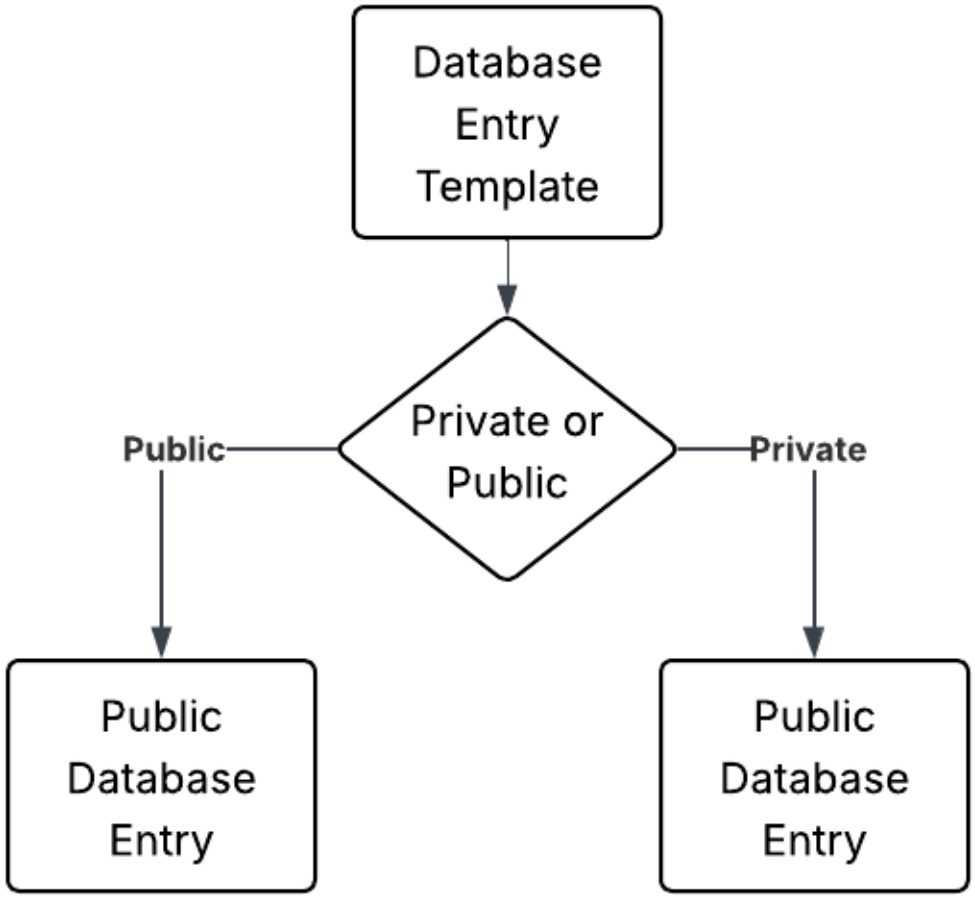
Iteration 2 Step 6

Element	Responsibility
Data Caching Layer	The data caching layer will ensure that a user's own database entries will remain available if the internet is unavailable. Given that some fishing spots are in more remote regions, this allows fishermen to access their data while out on a trip. (QA-3, CON-2, CRN-2)
Data Sharing Layer	The data sharing layer will allow users to elect to share their catches with other users on the service. They can choose whether or not to include information such as their real name when sharing. (QA-1, QA-2, CON-3)

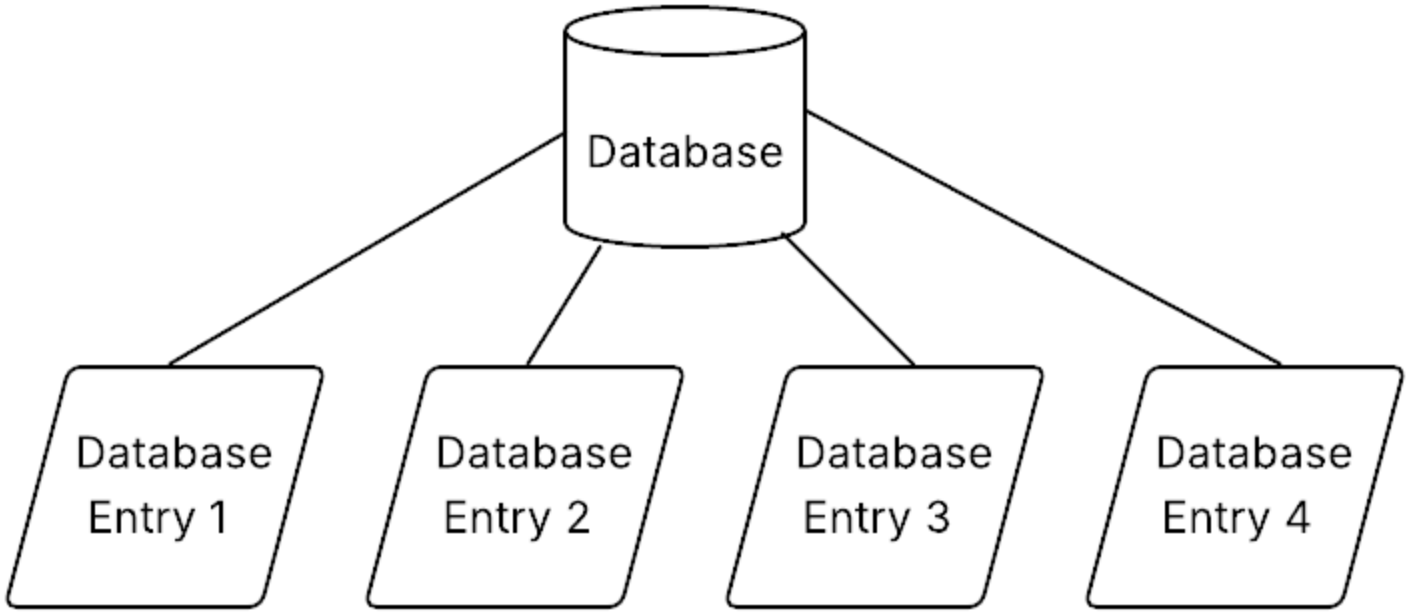
Layered Architecture Diagram



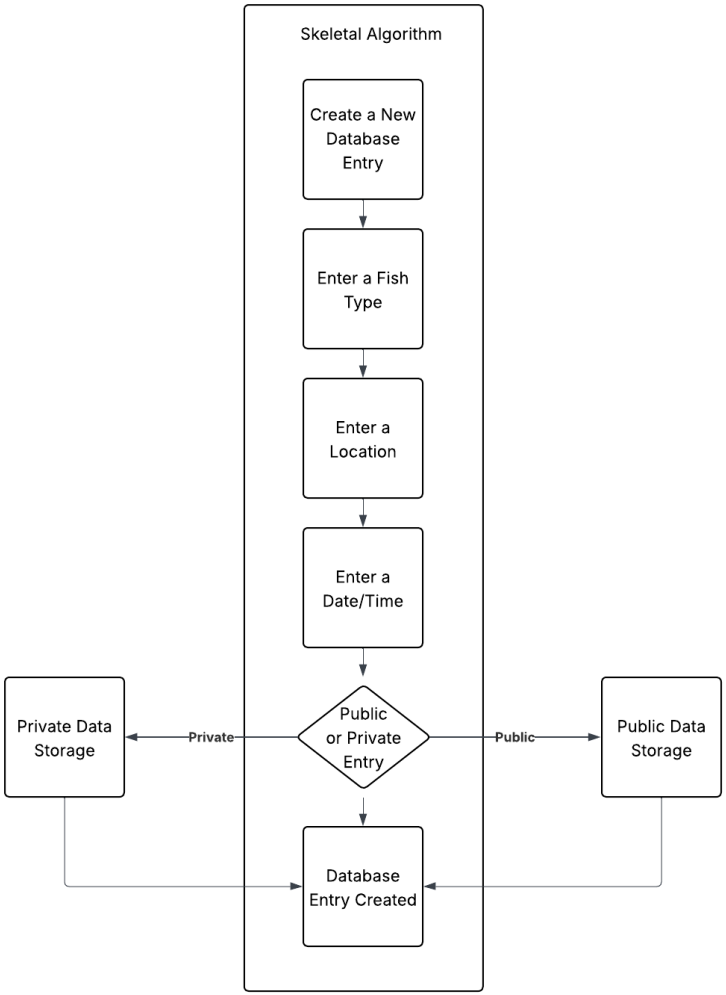
Creational Design: Prototype Diagram



Structural Design: Flyweight Diagram



Behavioral Design: Template Diagram



Iteration 2 Step 7

Not Addressed	Partially Addressed	Completely Addressed	Design Decisions Made During the Iteration
		UC-1	Used the Fat Two-Tier Client-Architecture to allow users to connect to the database and store/view catch information. Offline access in the form of data caching.
		UC-2	Textual Data Storage allows pro-fishermen to view important catch information to analyze and extrapolate data to better make decisions. Offline access in the form of data caching.
		UC-3	Textual and Image Storage allows more general wildlife entries to be made. Only manual entry is available for this feature. Offline access in the form of data caching.
		UC-4	Users can share their catch information with other users, with the option to turn them back to private at any time. Users can also choose to share some of their personal information such as their name but will never be forced to.

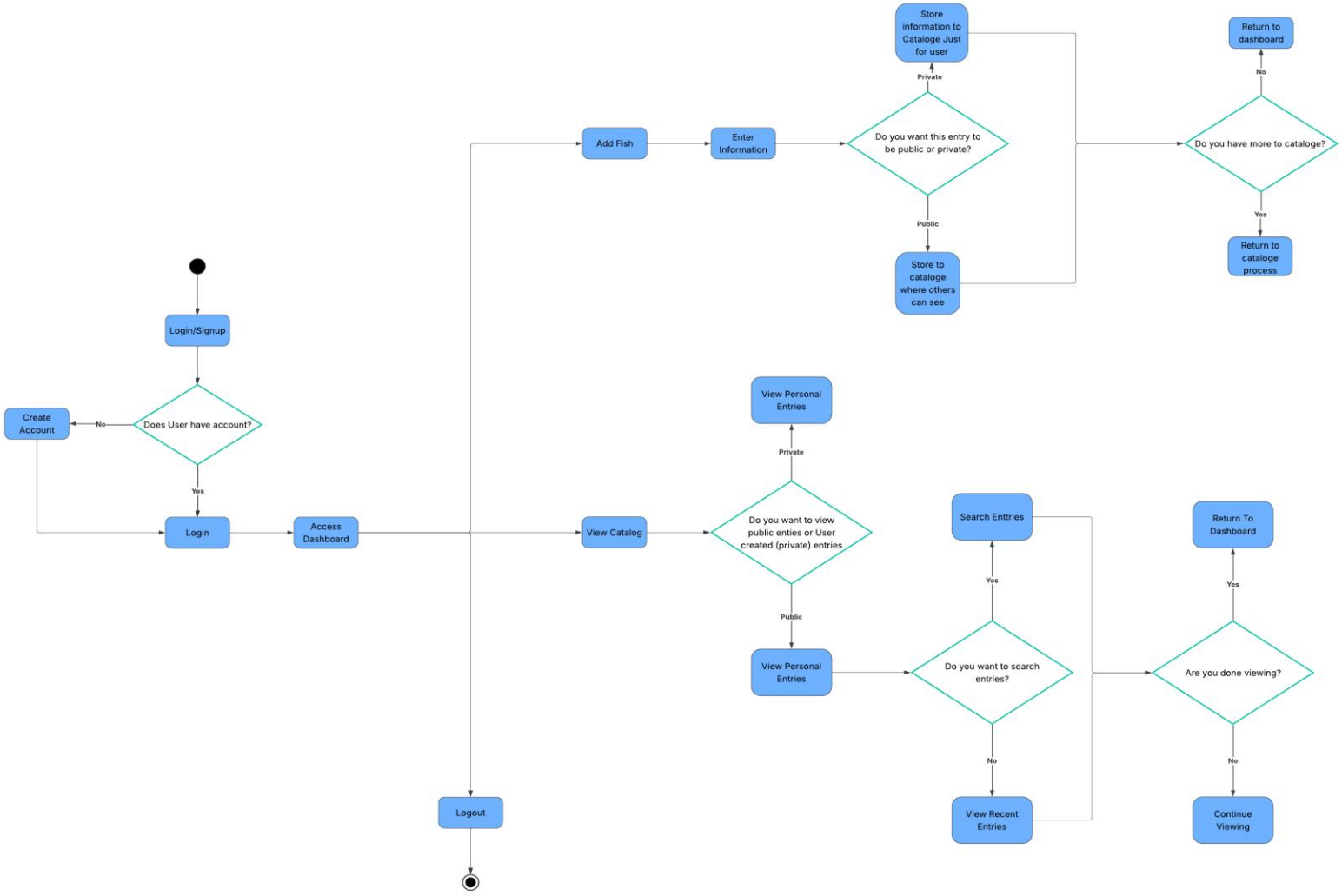
Iteration 2 Step 7 (2)

		QA-1	Users can share their catch information with other users.
		QA-2	Any personal information entered by the user will be stored securely on the server. Security features mentioned previously will keep information safe.
		QA-3	Offline caching now is fully implemented, allowing users to access certain database information while an internet connection is unavailable.
		QA-4	Due to the selected architecture, processing load is dependent on the client and not the server, therefore processing load should be well distributed geographically.
		QA-5	Due to the selected architecture, the database itself should not have much processing load relegated to it, therefore it will be speedy in its updates.
		QA-6	Text and Image Storage allows for long term data storage in the database.
		QA-7	Once the service enters production, a separate branch will be utilized for the implementation of new features.

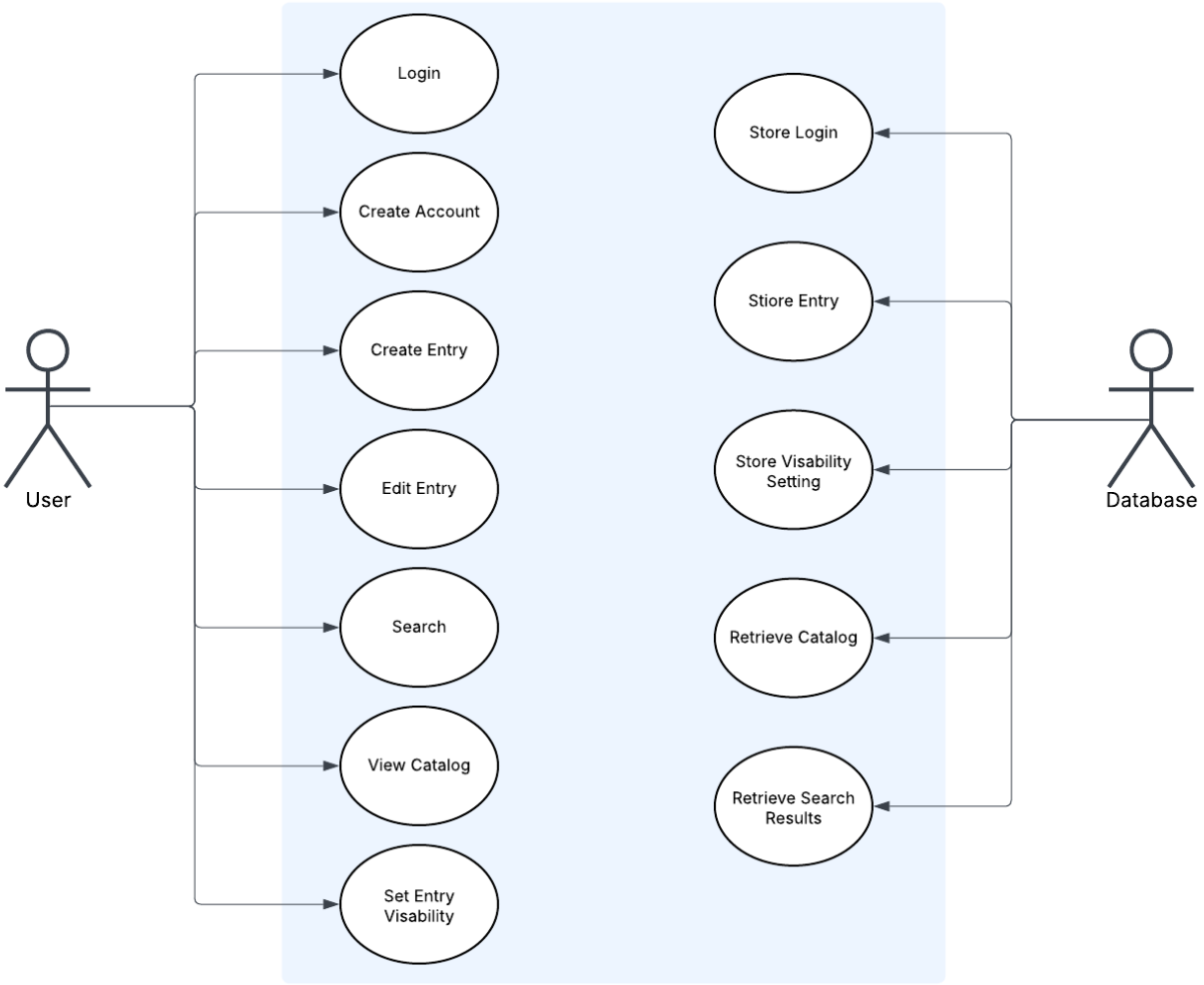
Iteration 2 Step 7 (3)

		CON-1	Some of the newly adopted architecture designs work well when handling large volumes of long-term data.
		CON-2	Now that data caching is implemented, offline access is available for certain database information.
		CON-3	Now that the data sharing layer has been implemented, shared data can be stored separately from private data.
		CRN-1	Users can create, read, update, and delete their own database entries.
		CRN-2	Now that data caching is implemented, offline access is available for certain database information.
		CRN-3	The adoption of the Fat Two-Tier Client-Server architecture means that new features and the expansion of current features can be implemented easier.

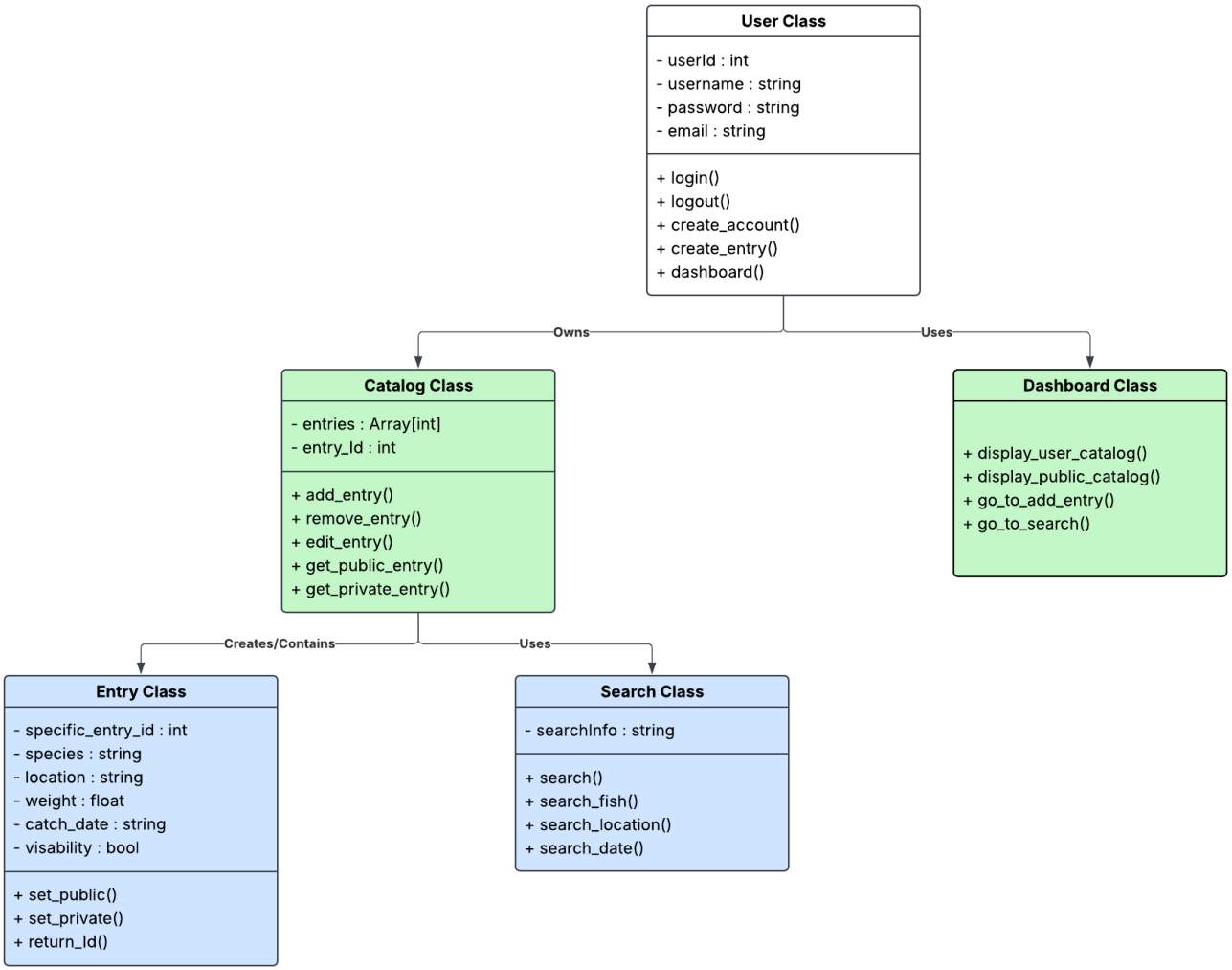
Activity Diagram



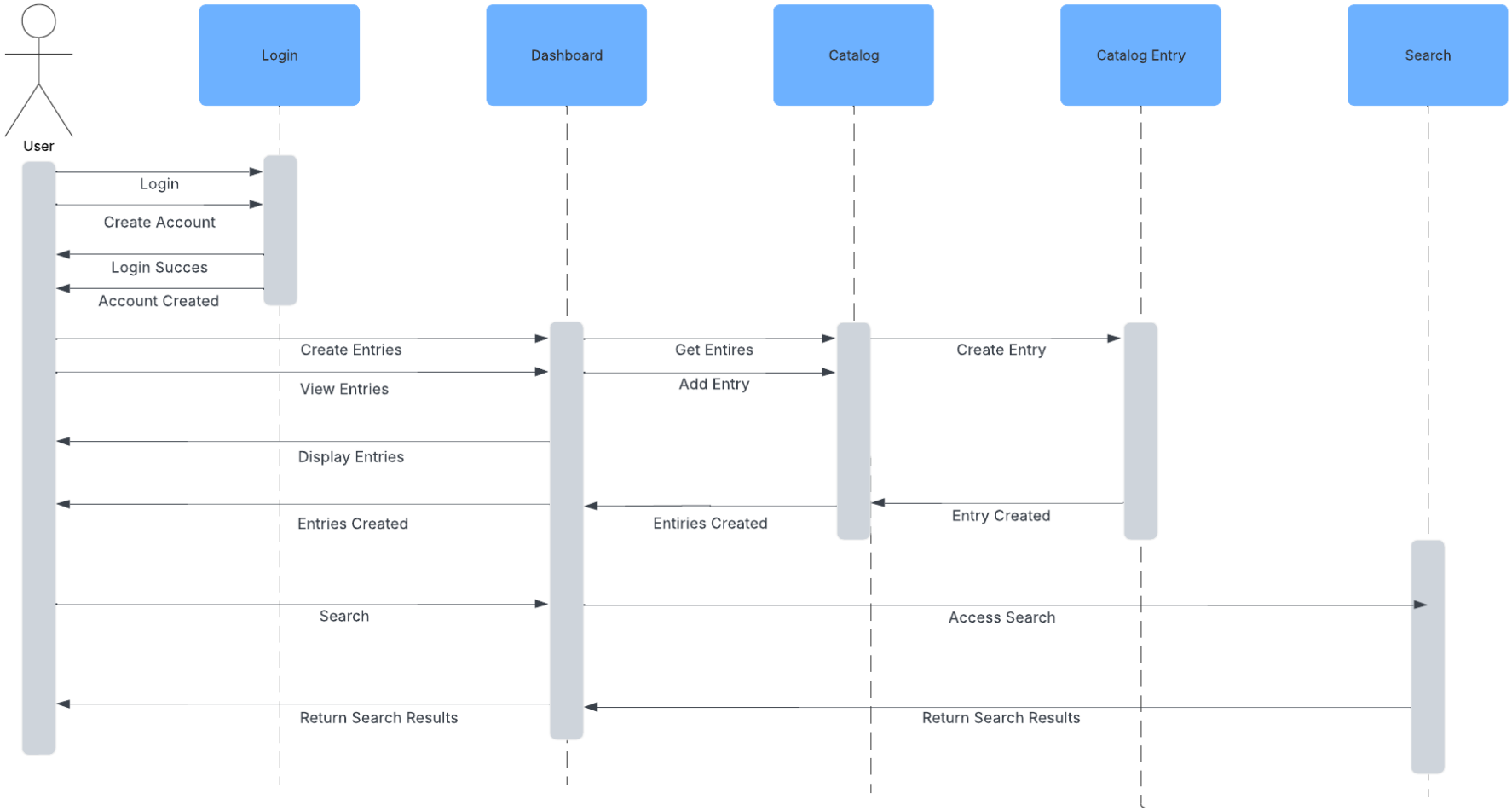
Use Case Diagram



Class Diagram



Sequence Diagram



University of Colorado
Colorado Springs

University of Colorado

Boulder | Colorado Springs | Denver | Anschutz Medical Campus