Lost Pet Finder chip/app: Design Assignment

CS 361: Software Engineering I

Group 16

Haya Ahmed
Ryan Alcorn
Marc Baiza
Shannon Farazi
Matthew Koenig

UML Class Diagram

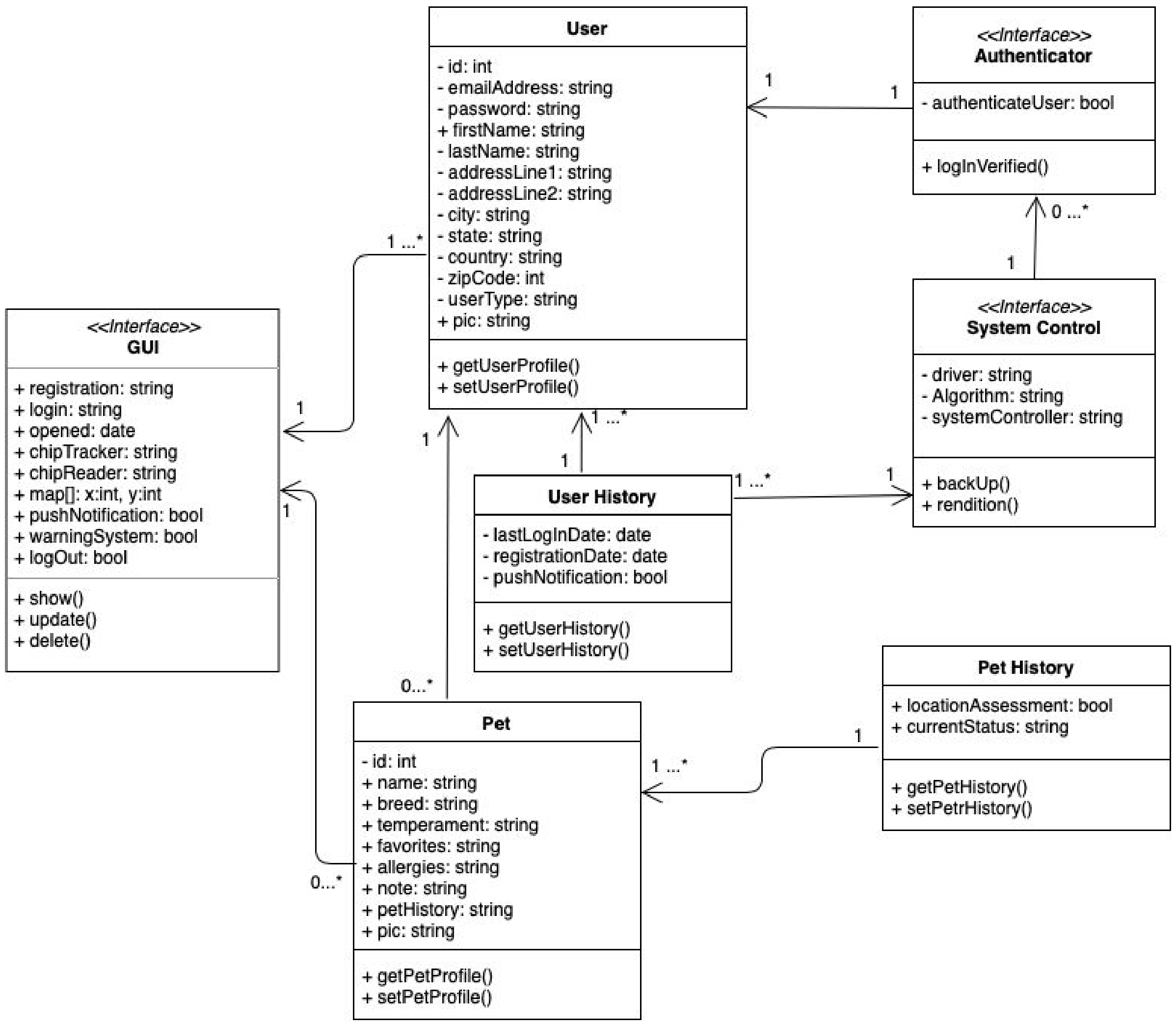

Figure 1: UML Class Diagram depicting simplified app connections.

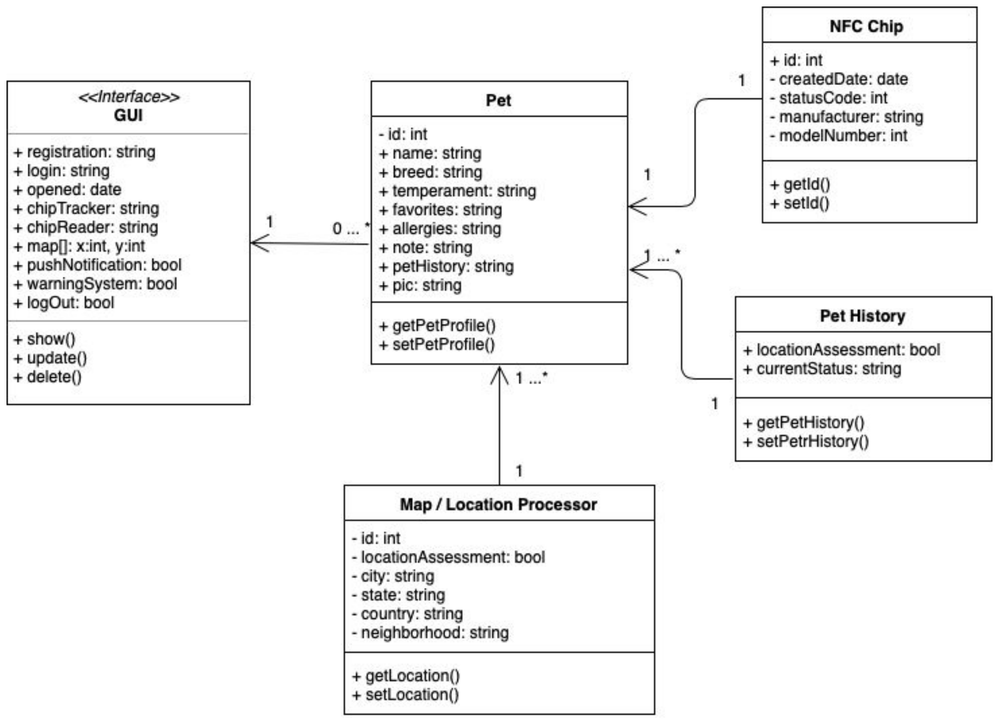

Figure 2: UML Class Diagram depicting simplified NFC Chip Tracking connections.

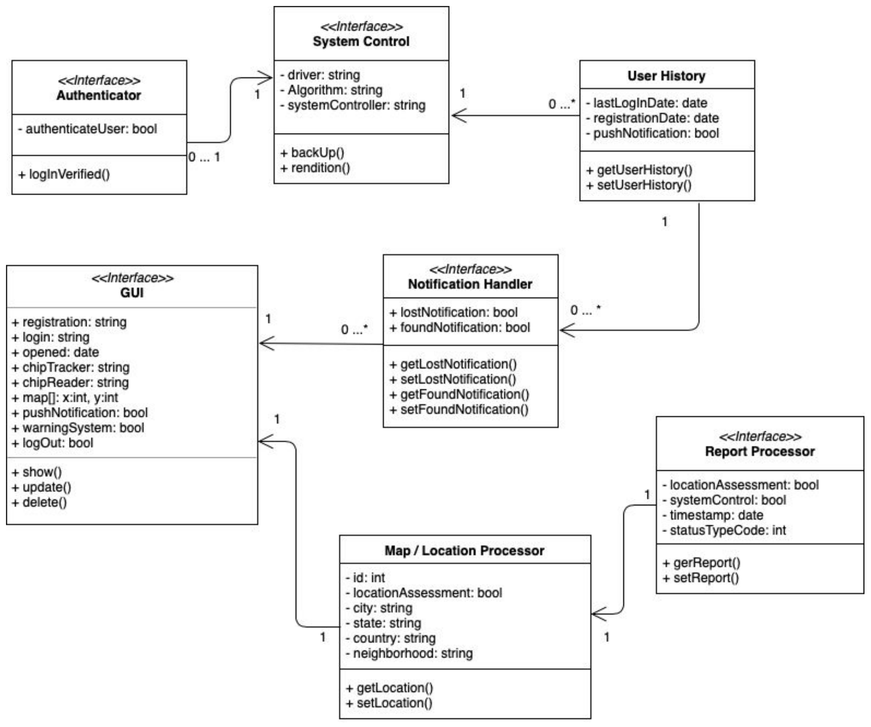

Figure 3: UML Class Diagram depicting simplified system control and notification handling connections.

Packaging The Implementations

The UML package diagram below depicts the complexity of our project modules with regards to coupling and cohesion.

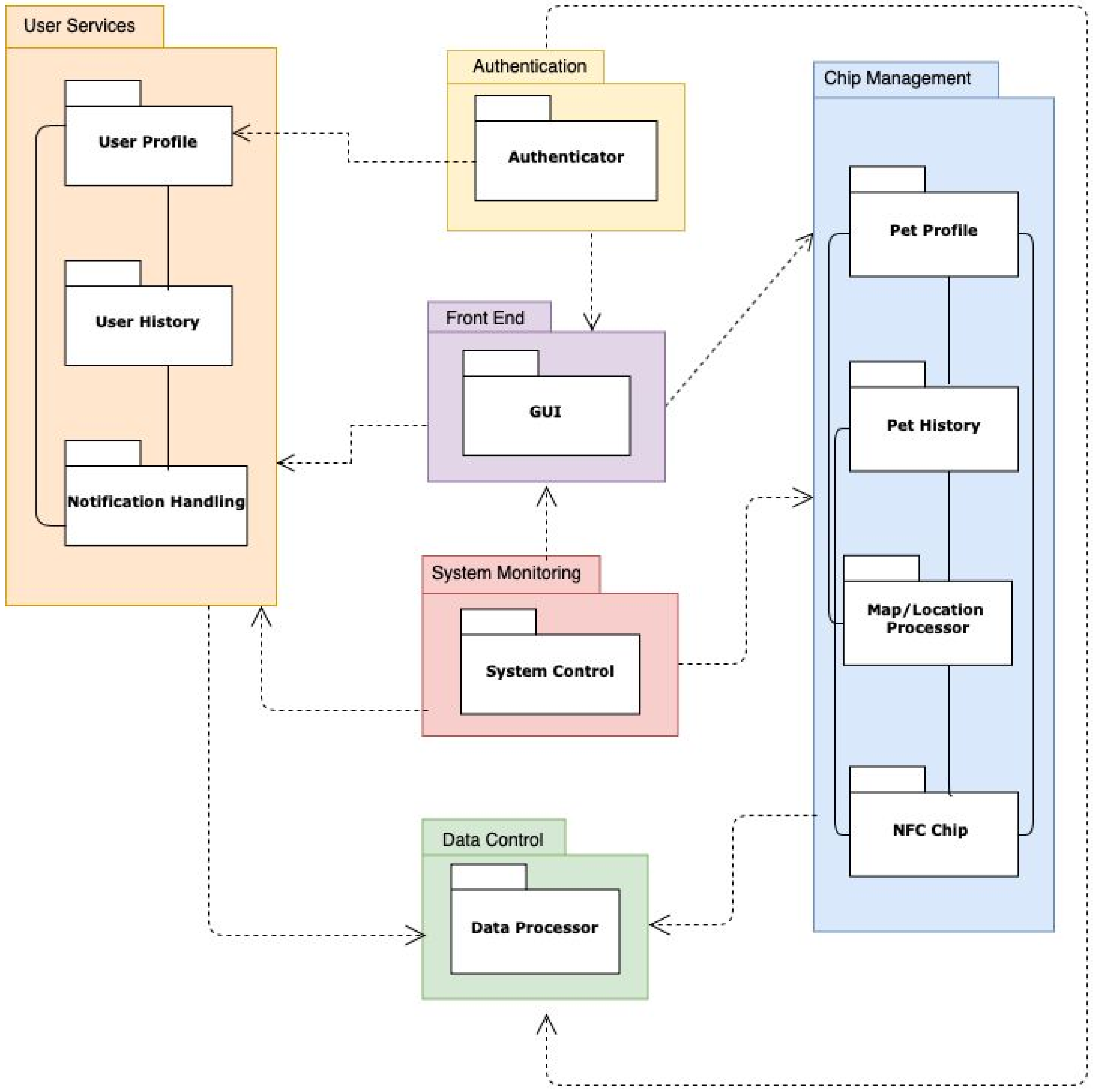

Figure 4: UML Package Diagram depicting simplified coupling and cohesion connections.

COUPLING

Coupling is when one module is dependent on another module's components. Since coupling reduces maintainability of the system, it is important to note that loosely coupled modules are preferred.

Content Coupling

- User services and the chip management modify other modules as they get updated by users.

Common Coupling

- The authenticator, GUI and system control at some point read and write the same data from and into user services respectfully.

Control Coupling

- The authenticator calls on the user profile.
- The chip management's components calls on data processor to access database.
- The system control calls on user services and chip management to control the flow of accurate data.

Stamp Coupling

- The User Profile provides structured data to the authenticator.
- User services provides structured data to the GUI.
- The chip management's components will provide structured data to GUI.
- The chip management's components will provide structured data to system control.

Data Coupling

- The User/Pet Profiles provide unstructured data to GUI.

Uncoupled Coupling

- This does not apply to our modules as they are set up according to the diagrams above.

COHESION

Cohesion focuses on how tightly the components of a module fit together. Since cohesion increases maintainability of the system, it is important to note that cohesive components within a module are preferred.

Functional/informational Cohesion:

- User Service - The purpose of the user service is to encapsulate the main user related data.
- Chip management - Package that handles data related to pet and pet tracking.
- Authenticator - Sole intention is to add security to the system.
- GUI - Provides the user interface for users to interact with the software.
- Data Processor - Sole intention is to manage the database of the system.

Communicational Cohesion:

- The components of user services use the same user data.
- The components of chip management handle the same user data.

Procedural Cohesion:

- This does not apply to our modules as they are set up according to the diagrams above.

Temporal Cohesion:

- This does not apply to our modules as they are set up according to the diagrams above.

Logical Cohesion:

- Either components of user services might be executed at a given time.
- Either components of chip management might be executed at a given time.

Coincidental Cohesion:

- This does not apply to our modules as they are set up according to the diagrams above.

Design Assessment

An incremental development approach is preferred when much of the system’s value resides in one section, and when one part of the system must be completed (logically) before another. Conversely, an iterative development approach is better when the whole system’s value is spread out over much of the system, and when the whole system needs to work at least a bit before you can build up.

For the purposes of our design, the incremental system is more appropriate, as the entire system must be working for the user to derive any benefit of its use. Each of the modules comprising the system works in sequential fashion, necessitating the completion of one before another is of use.

Of course, without a functioning GUI, there isn’t even a usable application, and from our customer's point of view, that is where the value lies. The design that is reflected in our Architecture where the Pet, NFC Chip, Pet History, Authenticator, System Control, User History, Notification Handler, Data Processor, and Map/Location Processor Classes all flow through the GUI Class.

Useful Design Patterns

Facade: The Facade design pattern would be useful to the system and the pattern relies on an object that provides a unified, high-level interface to a subsystem. The GUI of our system provides these features because it is a high-level interface that connects to other important subsystems in the system including the Notification Handler, Data Processor, System Controller, and Map/Location Processor. The user would only be able to see the GUI, but it would perform actions with all of these other systems to make the app functional. Some of the complex actions that the GUI would provide behind the scenes would be sending out the missing pet report, notifying the Notification Handler of the missing pet, and submitting user credentials to the System Controller to check login information. It would be a facade because the user will only see the flashy GUI and would perform app actions on there. They wouldn't see the inner workings of the processes that begin once they perform an action on the GUI.

Observer: Another design pattern that would be useful for the system is the Observer. The system that would be mainly affected in this design pattern would be the Map/Location Processor which displays missing pet tracking information/location to app users. The Map/Location Processor would be looking for a state change in the NFC Chip Tracker. This would most likely be driven by the user(owner) changing the status of their pet. The Map/Location Processor will then check with the NFC Chip Tracker to find out the location of the missing pet. It will continue to check in with the NFC Chip Tracker until the user(owner) indicates that the pet has been found.

Interpreter: The Interpreter would be another design pattern that would be useful for a couple components in the system. The two components that would function as an Interpreter are the Profile Recoder and Data Processor. Both of these components receive information from the GUI while it is performing certain functions. They both have to process this information and make it readable for other components in the system at lower levels.

Adapter: The Adapter would also be another useful design pattern for our system and it translates one system to another by wrapping. Our system has two separate pieces of technology in it that are required to function together. The system has the app which would be usually based in a phone and the NFC tracking chip. NFC chips already exist, but they do not currently implement the functionality that the app would require. The designer of the system would have to adapt the NFC system to function with the app and continue to update it based on future updates of the chip.

Builder: The Builder design pattern would also be useful for our system and it is a pattern that knows how to create a complex object. The database in our system is a component that will receive data of different types originally and create a complex object(an entry in the database) based off of this information. The database will instantiate new objects(entries in the database) when it receives information about new users or new pets filling them with new data.

Use Case Sequence Diagram

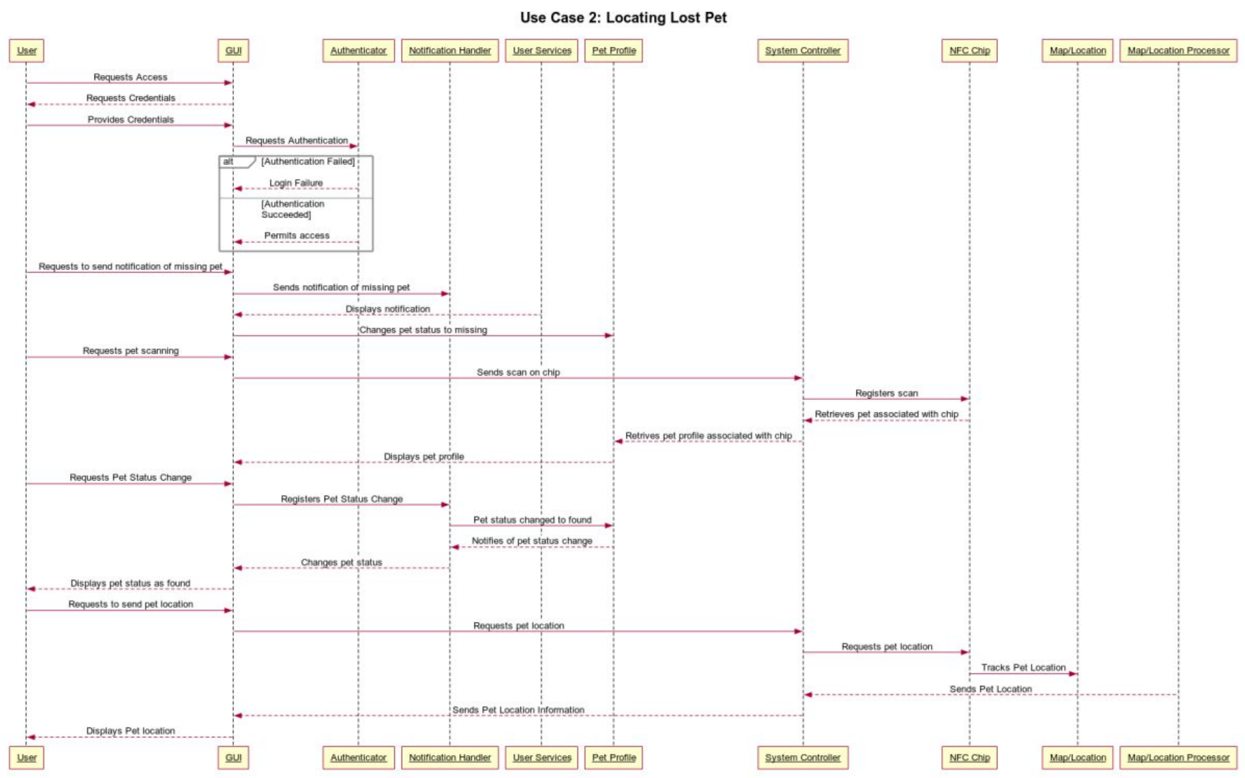

Figure 5: Use Case Sequence Diagram depicting Use Case 2 - Locating Lost Pet.

Interfaces

- **Driver for Chip**: Input: encrypted code that activates chip with NFC scan. Output: GPS signal that allows for location.
- **REST requests:** Input: User information required to login, User confirmation required to change any profile information, change setting, etc. (true/false displayed as yes/no). Input (Server): Change in server is then written depending on the response given by user(client input).

Inter-Package Interfaces:

- **Front End(GUI):** Input: Output information from Authentication, User Services, and Chip Management will be sent to the GUI as well as output information that flows from Data Control, and System Monitoring. Output: the display of the GUI that allows the user to interact with the system.
- **User Services:** Input: User authentication information from the authenticator is forwarded as well as any changes made to the users profile from the GUI. This includes profile changes, notification settings change, history clear, etc. Information/data flow from System Monitoring will also pass through Output: Once changes are made changes are reflected and sent back to the GUI and System Monitor.
- **Authentication** Input: Login information from GUI is sent to the authenticator. Output: Displays result notification to GUI whether or not information could be validated after compared to database of usernames/passwords.
- **Chip Management** Input: Information from GUI is passed on to chip management whether it be change in pet profile, pet history, location, etc. System Monitoring data will also be used to control location services, and chip data. Output: Will output information to data control which will hold locations of animals on the map, NFC chip information, time, and pet status.
- **Data Control** Input: Information from chip management which is interpreted such as GPS locations, NFC chip information, time, and pet status. Output: Some information such as time and pet location will flow back to the GUI through chip management to receive accurate updates on location/times.
- **System Monitoring** Input: Information from GUI, location services, and chip management will all flow to system monitoring. The system monitor manages the data that flows through all of the parts of the system above. Will control the information given to ensure proper outputs in GUI occur. Output: Data given by other parts will trigger system calls that ensure the proper function is occuring with the data being given.

Exceptions

Internal Exceptions

These would be exceptions relating to things like a failed user login or a non-matching chip, among others. For a failed user login, the user would be prompted to repeat the process and/or reset their information. For a non-matching chip, the underlying data would be stored for later use, and the user simply notified of the existence of the issue. A report would be issued to the platform's support team for review and resolution.

External Exceptions

These exceptions generally relate to network issues, which prevent a user from accessing the platform, either to notify other users of a lost or found pet. The system would attempt to connect to the network a fixed number of times. If the exception persists beyond that, the user would be notified of the error, and a report issued to the platform's support team for review and assessment. The user would also be asked to review their settings to ensure preconditions such as 'location services on' are satisfied.

Other Exceptions

These exceptions focus on things such as the NFC scanner failing to sufficiently or accurately read the chip in the pet. Much in the same way other platforms handle bad scans (e.g. Google credit card, mobile checking, etc) the use will be given the option of entering whatever data is available on the pet's tag manually. As with internal and external exceptions, a report will be issued to the platform's support team for review and assessment.

Team Member Contributions:

- Haya: Use Case Sequence Diagram
- Matt: Design Assessment and Exceptions
- Marc: Interfaces
- Ryan: Useful Design Patterns
- Shannon: UML Class Diagram and Packaging the Implementations
- All: Strategy and Planning, Editing