The FindYourPet\_BACKEND is a Spring Boot application that provides server-side functionality to the Find Your Pet website. This application was created as a part of the Databases category for the CS 499 course at SNHU.

This artifact is included in my ePortfolio because it demonstrates my ability to design databases and create a back-end server with API endpoints that allows for CREATE, READ, UPDATE, DELETE operations of http protocol. It showcases my understanding of system design patterns and good API design. This Spring Boot application creates two tables in a PostgreSQL database and stores user and animal information. It follows the three-layer architecture, Model-View-Controller, and follows industry standard coding practices such as password hashing, using Object Relational Mapping via Data Transfer Objects and inline comments to make the code readable. This application was developed from scratch and does not have a prior artifact to improve upon. Compared to my previous backend work, the improvements I would like to showcase are good documentation of code via inline comments and good method and variable names that follows camel casing.

Yes, I have met all course outcomes with this artifact:

* Employ strategies for building collaborative environments that enable diverse audiences to support organizational decision-making in the field of computer science.
* Develop a security mindset that anticipates adversarial exploits in software architecture and designs to expose potential vulnerabilities, mitigate design flaws, and ensure privacy and enhanced security of data and resources.  
  + - Design, develop, and deliver professional-quality oral, written, and visual communications that are coherent, technically sound, and appropriately adapted to specific audiences and contexts.
    - Demonstrate an ability to use well-founded and innovative techniques, skills, and tools in computing practices for the purpose of implementing computer solutions that deliver value and accomplish industry-specific goals.

During the process of developing this artifact, I understood the importance of good relational database design, I also learnt the value password hashing brings in terms of security. While developing, I introduced a lot of checkers that validate incoming data which lead me to adopt data validation checkers in all my future projects. Some challenges I faced were relational database design, as I wanted to keep an animal in the database for other users even after one user removed it. I remedied it by allowing users to soft delete an animal from their profile only, and an endpoint that can run periodically, which deletes any animals in the table if they are not being used by any users. I also faced some issues with DTO mapping, as I was using the map struct library to generate boilerplate codes at first, but due to my decision to use BCrypt for password hashing, I had to take a manual approach and inject the hashing methods within my mapper.

Artifact Documentation:

Installation: please ensure PostgreSQL and Java are installed in the local machine. Either replace the environmental variables in the application.properties file with PostgreSQL username and password or set system wide env variables that match the file. Once the server is running, API endpoints can be tested via Insomnia, Postman or similar software.

The current app uses a create-drop approach where it drops all tables on app stoppage, for data persistence change line 6 in application.properties to spring.jpa.hibernate.ddl-auto=update

GitHub: <https://github.com/MKHLink/Find_Your_Pet_Backend>

API documentation:

POST: <http://localhost:8080/user/register>

POST: <http://localhost:8080/user/login>

DELETE: <http://localhost:8080/user/delete/1> (replace 1 with animal id)

Expected json request body for the above endpoints:

{

“email”: “johndoe@gamilcom”,

“password” : “12345abc”

}

PUT: <http://localhost:8080/user/save-animal>

Expected json request body for the put request:

{

"user": {

"email": "johndoe@gmail.com",

"password": "123142acv"

},

"animal": {

"image": "https://example.com/dog.jpg",

"name": "Sif",

"species": "Dog",

"breed": "Golden Retriever",

"city": "New York",

"contact": "123-456-7890",

"status": "Available"

}

}

GET: <http://localhost:8080/animal>

DELETE: <http://localhost:8080/animal/1> (replace 1 with animal id)