

Florida International University
Knight Foundation School of Computing and Information Sciences

sSoftware Engineering Focus

Final Deliverable

Project Title:
FloriDex

Team Members: Lucas Rodriguez, Amaury Pardo, Erick Sanchez, Diego Angueira,
Michael Rosales

Instructor: Masoud Sadjadi

The MIT License (MIT)
Copyright (c) 2016 *Florida International University*

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.

Abstract

This document presents the information necessary to gain a good understanding of what the FloriDex project is about and what it aims to be. Currently, the first page prototypes are complete using a Model-View-ViewModel architecture and the database is created with their lambda functions.

Table of Contents

INTRODUCTION	5
CURRENT SYSTEM	5
PURPOSE OF NEW SYSTEM	5
USER STORIES	
IMPLEMENTED USER STORIES	6
PENDING USER STORIES	8
PROJECT PLAN	
HARDWARE AND SOFTWARE RESOURCES	9
SPRINTS PLAN	10
<i>Sprint 1</i>	10
<i>Sprint 2</i>	10
<i>Sprint 3</i>	10
<i>Sprint 4</i>	10
<i>Sprint 5</i>	10
<i>Sprint 6</i>	10
<i>Sprint 7</i>	11
SYSTEM DESIGN	
ARCHITECTURAL PATTERNS	11
SYSTEM AND SUBSYSTEM DECOMPOSITION	12
DEPLOYMENT DIAGRAM	14
DESIGN PATTERNS	14
SYSTEM VALIDATION	15
GLOSSARY	16
APPENDIX	17
APPENDIX A - UML DIAGRAMS	17
APPENDIX B - USER INTERFACE DESIGN	21
APPENDIX C - SPRINT REVIEW REPORTS	24
APPENDIX D - USER MANUALS, INSTALLATION/MAINTENANCE DOCUMENT, SHORTCOMINGS/WISHLIST DOCUMENT AND OTHER DOCUMENTS	26
REFERENCES	30

INTRODUCTION

The FloriDex, or Florida Fauna & Flora, is a Kotlin-based mobile application designed to provide information and insights into the diverse range of wildlife species and flora found in Florida. Whether you're a nature enthusiast, a student, or a curious traveler, this app is your ultimate companion to explore and learn about Florida's fascinating fauna and flora.

Current System

FloriDex can display a list of creatures and show details about each one, including their sound. There are built-in comments put up to describe opinions about the creatures. Basic authentication is implemented for users to have their own account within the app, and they can see their profile while having the option to sign out. Overall, the project contains basic functionality.

Purpose of New System

FloriDex drives on having community engagement and user experience. This can be achieved by having the option to create creature entries and comments about them. Blogs can also appear in the form of articles with new posts appearing occasionally. Finally, users should have the option to customize their profile information and manage their entries and comments.

USER STORIES

The following section provides the detailed user stories that were implemented in this iteration of the mobile application. These user stories served as the basis for the implementation of the project's features. This section also shows the user stories that are to be considered for future development.

Implemented User Stories

User Story 1: As a developer, I would like to create a prototype for the login and register pages, so that we would have the first working design after the wireframe

User Story 2: As a developer, I would like to create a prototype for the creature list page, so that we would have the first working design after the wireframe

User Story 3: As a developer, I would like to create a prototype for the creature detail page, so that we would have the first working design after the wireframe

User Story 4: As a developer, I would like to create a prototype for the profile page, so that we would have the first working design after the wireframe

User Story 5: As a developer, I would like to learn Kotlin and Android Studio, so that way I can get up to speed with the project.

User Story 6: As a developer, I would like to update the login and register prototype, so that we would have 2 pages together

User Story 7: As a developer, I would like to update the creature list prototype, so that we would have a more improved version of it

User Story 8: As a developer, I would like to update the creature detail prototype, so that we would have a more stable version of the 3 pages

User Story 9: As a developer, I would like to update the profile prototype, so that we would have a more stable version of the page

User Story 10: As a developer, I would like to begin work on a prototype of the settings/preferences page, so that way it will be ready to be wired with the other menus in the future.

User Story 11: As a developer, I would like to set up a database, so that we can connect it to our application

User Story 12: As a developer, I would like to connect the database to the creature list, so that we can sync the list of data from the database

User Story 13: As a developer, I would like to connect the database to the creature's description page, so that we can sync data about the creature from there

User Story 14: As a developer, I would like to connect the database to the profile page, so that user info is synced

User Story 15: As a developer, I would like to connect the database to the settings page, so that user input can overwrite and save to the database

User Story 16: As a developer, I would like to connect the database to the login and register page, so that we can sync the user's authentication

User Story 17: As a developer, I would like to implement an API connecting to the database, so that we can better the data sync

User Story 18: As a developer, I would like to connect the database API to the Android project and utilize its data, so that we can all rinse out API bugs

User Story 19: As a developer, I would like to connect the creature list to the description page, so that each creature has its navigation and info

User Story 20: As a developer, I would like to connect the login/registration page to the creature list, so that users can have an authentication process result

User Story 21: As a developer, I would like to connect the creature list and creature description to the profile page, so that users can have easy access to their profile

User Story 22: As a developer, I would like to connect the profile page to the settings page, so that users can have access to account customization

Pending User Stories

User Story 23: As a developer, I would like to create an implementation for changing username and password, so that the settings page has a better UX

User Story 24: As a developer, I would like to create an implementation to post comments to creature descriptions, so that community engagement increases

User Story 25: As a developer, I would like to create a blog, so that users can have more reasons to interact with the app

User Story 26: As a developer, I would like to update the location tab in the description page, so that users can visually see in the map where creatures are

User Story 27: As a developer, I would like to create an implementation to change profile pictures, so that users can better represent themselves

User Story 28: As a developer, I would like to create an implementation to post creature entries, so that the app can promote individual enrichment

User Story 29: As a developer, I would like to set an implementation to favorite creature entries, so that users can reference them when necessary

PROJECT PLAN

This section describes the planning that went into the realization of this project. This project incorporated the agile development techniques and as such required the sprints to be planned. These sprint plannings are detailed in the section. This section also describes the components, both software and hardware, chosen for this project.

Hardware and Software Resources

The following is a list of all hardware and software resources that were used in this project:

- **Android Studio Integrated Development Environment (with built-in emulator)**
- **Amazon Web Services Relational Database Service (MySQL)**
- **Amazon Web Services Lambda Service (JavaScript)**
- **Amazon Web Services API Gateway**
- **GitHub**
- **MySQL Workbench**

Sprints Plan

Sprint 1

Unless someone in the team is not familiar with the Kotlin programming language and Android Studio, everyone in the team would use available wireframes and diagrams to come up with a prototype for the app's pages necessary for basic functionality. Otherwise, give them a week or two to catch up with the technology with assistance for project contribution. (User Stories 1 – 5)

Sprint 2

If someone in the team took last sprint to get familiar with the technologies, he or she would start the prototype for their assigned application page. Other members of the team would continue their app page prototypes for a better version. (User Stories 6 – 10)

Sprint 3

One person with familiarity of database design would create a remote database containing tables and data ready for use with the project. Meanwhile, the other members continue working on their prototypes until the database is ready for use. (User Stories 11 – 16)

Sprint 4

Once the remote database is set and ready, every team member would come up with an API using AWS lambda and API Gateway for use with the project. (User Story 17)

Sprint 5

Every team member would continue to build their API implementation and ensure there are no issues between the service and Android Studio. (User Story 18)

Sprint 6

Determining that the API is difficult to implement, every team is given another sprint or two to finish the API implementation. Any member that finishes can start implementing connections to the app's other built pages. (User Stories 18 – 22)

Sprint 7

While every team member is given this sprint to finish the page API and connections, everyone is tasked with creating a poster for the senior design showcase. The team lead or any volunteer is also tasked with creating the presentation slides, project documentation, and videos introducing and demoing the project. Note that the demo requires every team member finishing their pages. (User Stories 18 – 22 & Tasks 1 – 4)

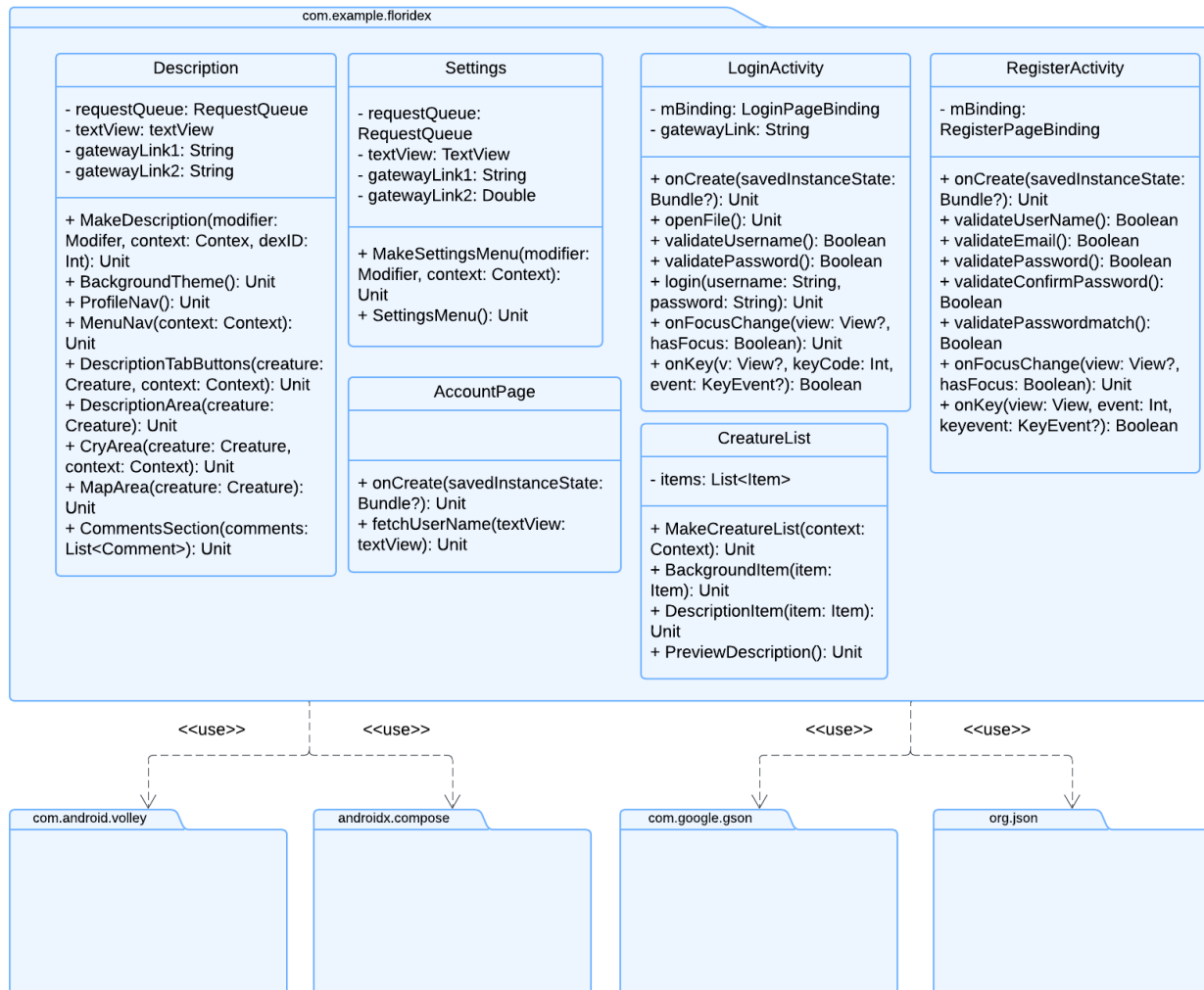
SYSTEM DESIGN

This section contains information on the design decisions that went into this project. The architecture patterns are outlined and explained. The entire system is shown in a package diagram and the subsystems are explained. Finally, the design patterns used in the project are discussed.

Architectural Patterns

Since the project would rest upon navigating between pages and using both wildlife info and user info from a database to update the app's display, a Model-View-ViewModel is used for maintainability purposes. The view model would turn especially more useful when more features handling user input get added and the display changes often.

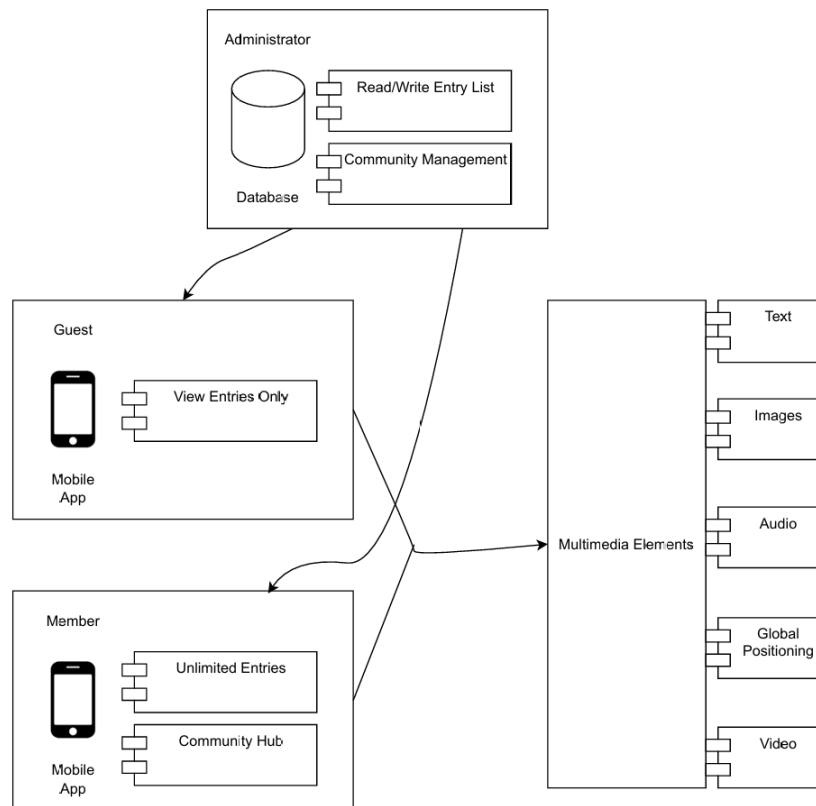
System and Subsystem Decomposition



- `com.example.floridex`: The main package runs the project and all the classes within it.
 - `Description`: Sets up the description page by creating a request for both the wildlife table and the comments table, then creates a list containing the creature filtered by its id. The creature's info is structured into the info tab containing the picture and facts and the cry tab containing the sound. Navigation to the profile page is performed as well.

- Settings: Sets up the settings page by creating a request for the user table then creates the user list containing the user filtered by its email. The info structures the page with the edit username button, change password button, delete account button, and sign out button going to the login page
- AccountPage: Sets up the profile page by creating a request for the user table then creates the user object filtered by its email address. The username then gets fetched to help display with the profile picture and its recent content section.
- LoginActivity: Sets up the login page by creating a request for the user table then creates a list containing every user in the database. The page is structured with a username and password field along with the login button. The input fields match with the list fields until 1 user gets identified. Page navigation to the profile or settings passes through with that user object.
- RegisterActivity: Sets up the register page by first structuring the layout with the fields like username, password, and email necessary for a new user instance. The register button would add the new instance to the database's user table and navigates to the profile or settings.
- CreatureList: Sets up the creature list page by creating a request for the wildlife table then creates a list containing every creature in the database. The layout fits every instance in the list as a box containing brief info with a picture. The box serves a button to navigate to the description page with the selected creature's id.
- com.android.volley: The network package that interacts with data transferred within the web. Requests obtain the data through an https site and return a response, normally to be parsed as a JSON object.
- org.json: The json package builds upon the JSONObject and all related data types and methods. For instance, The JSONObject parses the data objects based on its key-value structure surrounded by curly braces ({}).
- com.google.gson: JSON parser and converter package that interacts with data classes and collections. Gson goes through each data or object element and performs the necessary conversion.

Deployment Diagram



Design Patterns

- **Model-View-ViewModel (MVVM)** – The ViewModel handles the data passed from the database and adapts it to the view's UI accordingly.
- **Command** – The request gets encapsulated into volley's string request object to be added into a request queue and parameterized with the request's response data
- **Façade** – A simplified UI is implemented within a complex application, allowing users to perform easy inputs and have a greater UX

SYSTEM VALIDATION

To ensure the quality and functionality of the project, we implemented a comprehensive testing strategy that encompassed bottom-up, UI, and module testing.

Bottom-up Testing

The team thoroughly tested the individual components and modules of the application. This involved creating each function and class and isolating them to verify their correct behavior in isolation. By focusing on the building blocks, I could identify and resolve issues early in the development process, preventing them from propagating to higher levels of the application.

UI Testing

Once the individual components were validated, the team shifted their focus to the user interface. We conducted a series of UI tests to ensure that the application's visual elements and user interactions were consistent, intuitive, and responsive. These tests covered aspects such as button clicks, screen transitions, form validation, and error handling. By simulating real-world user scenarios, everyone could identify and rectify any usability or visual design flaws.

Module Testing

Finally, the team integrated the tested modules and components to perform module-level testing. This involved simulating real-world workflows and scenarios to assess the overall functionality and performance of the application. We used UI testing after integration to cover various user paths, edge cases, and potential error conditions. By thoroughly testing the integrated modules, we could ensure that the application functioned as expected and met all the specified requirements.

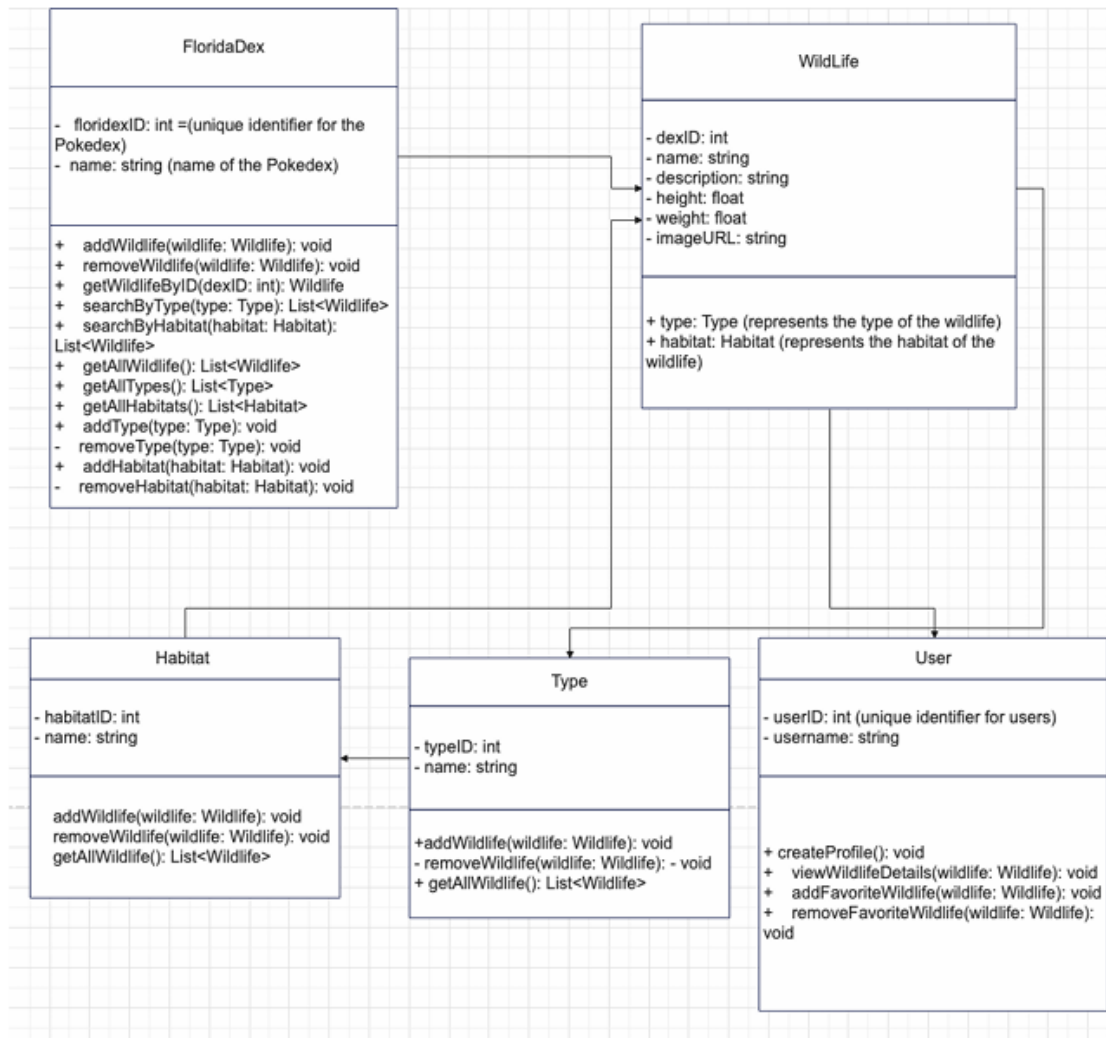
GLOSSARY

- **Model-View-ViewModel (MVVM):** An architectural pattern used to separate application logic and bind the user interface to an app's view model
- **Kotlin:** A general-purpose programming language created by JetBrains starting in 2011. It was designed as a simpler and more concise alternative to Java
- **Authentication:** A way for people to gain access to a system using their credentials
- **User Experience (UX):** How a user engages with the application by customizing and adding to their information
- **User Interface (UI):** How a user interacts with the application through the app's objects (i.e., button)
- **User Story:** A goal typically set by an organization's product owner or stakeholder
- **Prototype:** A test version of a product for progression purposes
- **Wireframe:** A product's outline sketch or blueprint
- **Database:** A group of tables containing organized data
- **Application Programming Interface (API):** A service that communicates with different software applications used to achieve better data integration and functionality
- **Android Studio:** An integrated development environment used to create and test android applications
- **Integrate Development Environment (IDE):** A tool that allows developer to implement, debug, and run software applications
- **Package:** An encapsulation of classes and interfaces for code organization
- **Class:** An encapsulation of organized code normally enclosed in a file

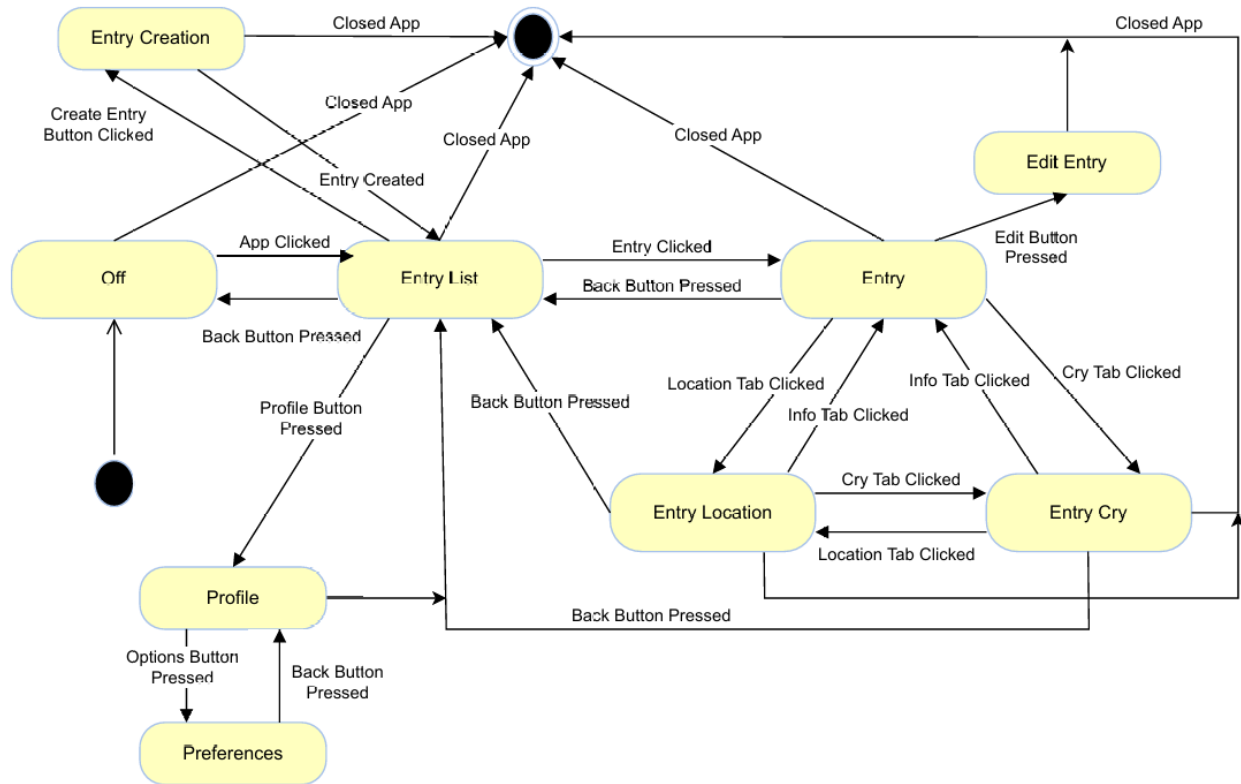
APPENDIX

Appendix A - UML Diagrams

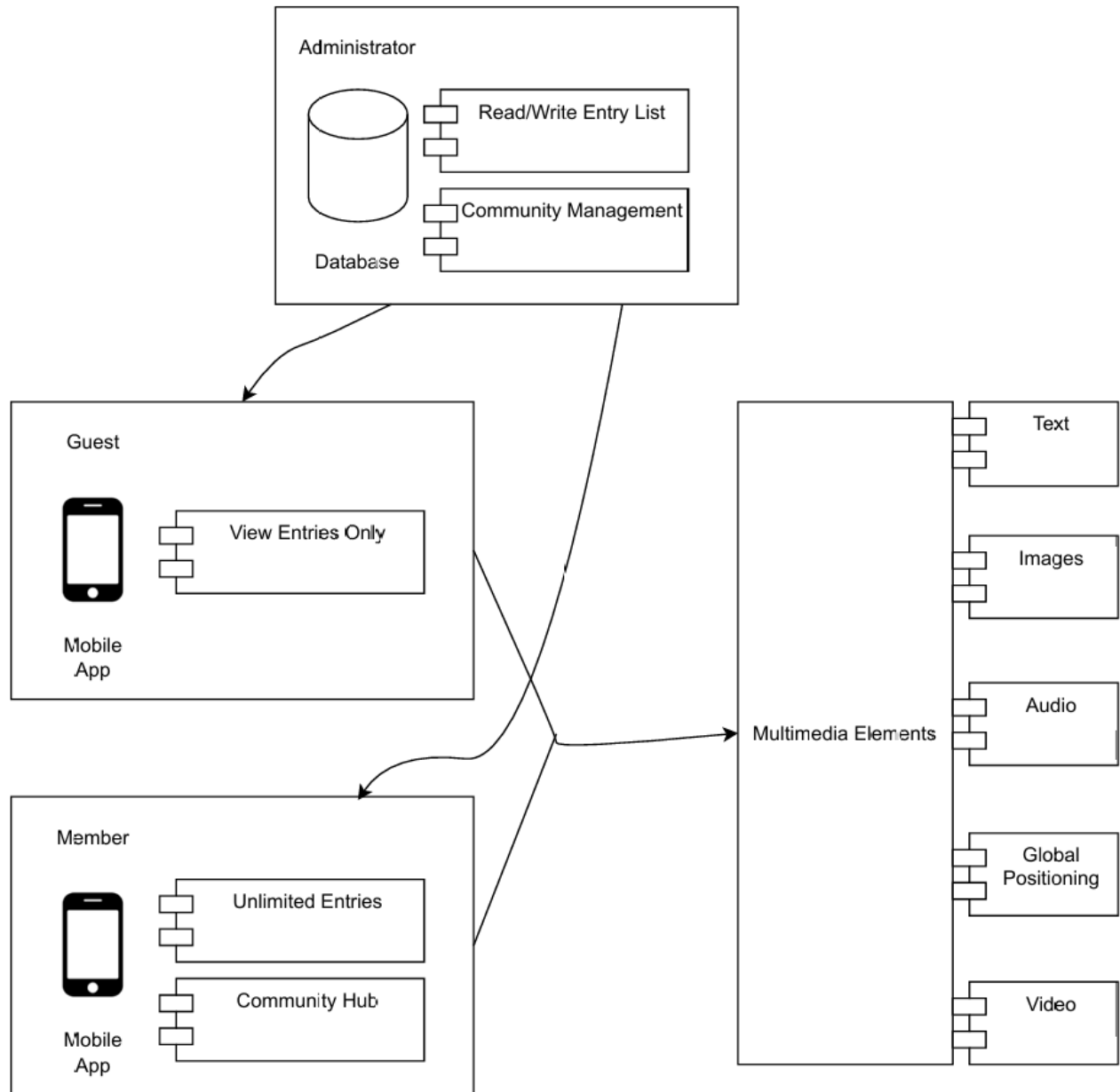
Class Diagram



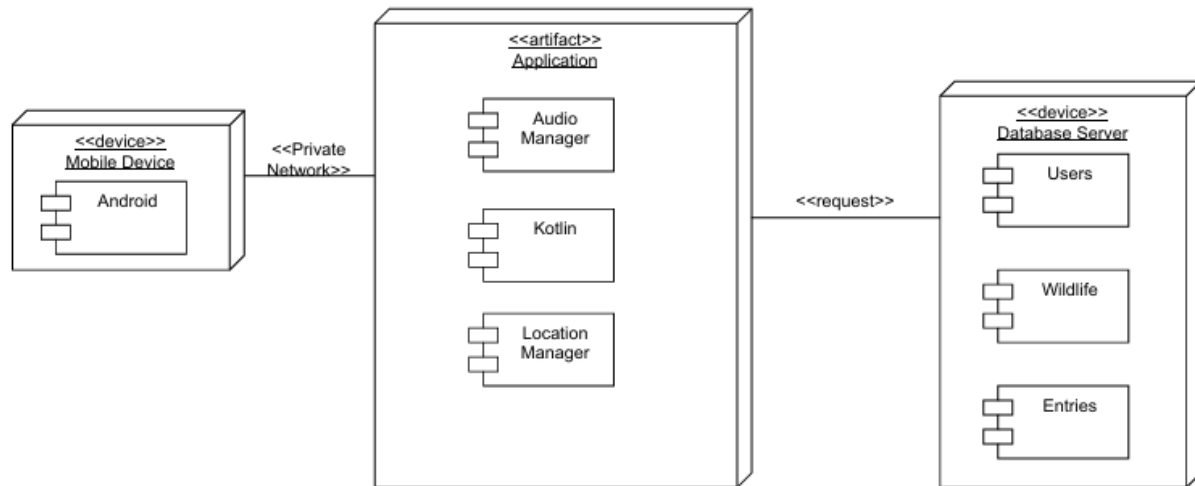
Statechart Diagram



Deployment Diagram



Architecture Diagram



Appendix B - User Interface Design

Login/Register Page

10:30

FloriDex

Email

Password

Login

Don't have an account?

Register

Account Setup

Username

Email

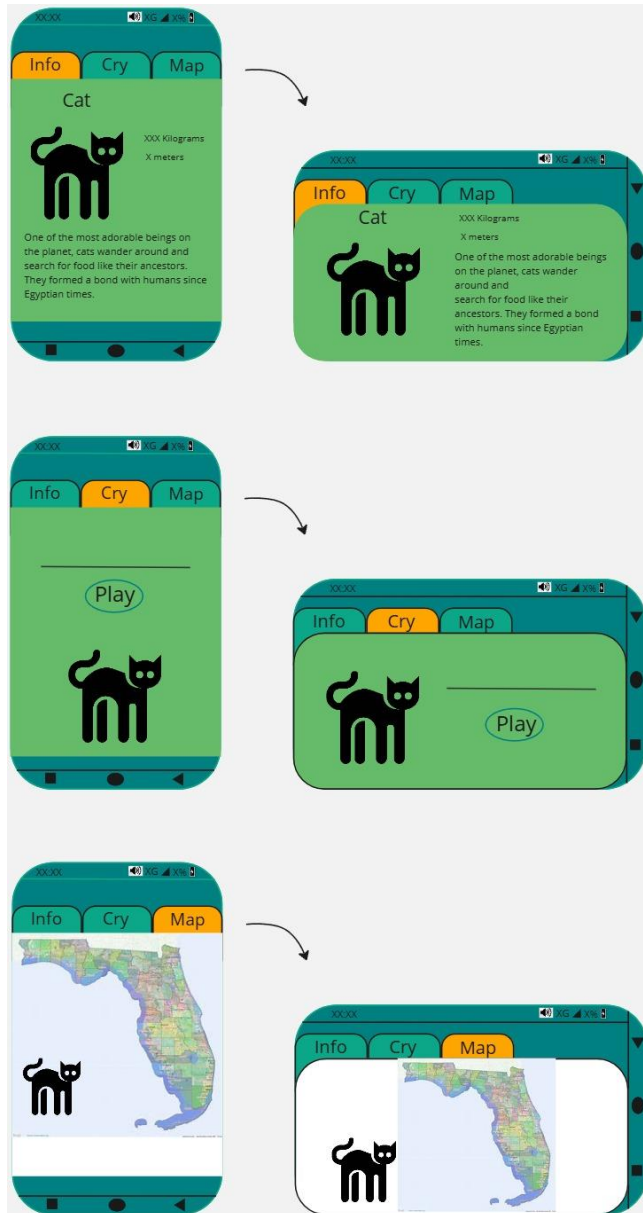
Password

Register

Profile Page



Description Page



Appendix C - Sprint Review Reports

Sprint 1

- All prototypes did not have issues but did not function completely
- The following suggestions were made for next sprint:
 - Social Profile page could have their own settings
 - Creature Description would need the actual creatures instead of placeholder
 - Creature list needs the list of creatures from a database
 - Login and register pages need to link to the social profiles

Sprint 2

- The following page layouts functioned without issues
 - The creatures' description layout
 - The basic settings
 - The profile layout
 - The login/registration layout
 - The creature list layout
- A database was suggested for creation and setup

Sprint 3

- The following issues arose when using AWS:
 - AWS database not properly connecting to Android Studio directly
 - AWS Lambda function API not returning any data for the description page
- Because of the issues, a working database was suggested for project progression

Sprint 4

- The following pages did not have a working database API connection:
 - The creature list page
 - The profile page
 - The login and register pages
 - The settings
- A working API connection was made for the description page, but parameters were needed for the page's goals of having specific creatures

Sprint 5

- Description page has a response error 403 caused by a faulty API gateway
- The following are suggestions for next sprint:
 - Creature List needing updated list syncing
 - Login/registration pages need creature list page connections
 - Settings page could manipulate profile data
 - Profile should connect to the settings page and grab user data

Sprint 6

- The description page's basic functionality and API implementation works without any issues
- The profile page's API implementation is not functional
- The following pages needed more sprucing up for complete functionality:
 - Settings
 - Login and Register Pages
 - Creature List

Appendix D - User Manuals, Installation/Maintenance Document, Shortcomings/Wishlist Document and other documents

User Manual

Introduction

Welcome to the FloriDex User Manual! This guide will help you navigate and utilize the features of our mobile application.

Getting Started

- 1. Download and Install:**
 - Download the FloriDex app from the GitHub repository.
 - Install the app on your compatible mobile device.
- 2. Create an Account:**
 - Open the app and tap on "Register"
 - Provide your email address, password, and other required information.
- 3. Log In:**
 - Enter your email address and password.
 - Tap on "Log In."

Main Features

Feature 1: Comprehensive Database

- **How to Use:**
 - Log in to the app
 - Browse through the list of species
- **Benefits:**
 - More engagement with the variety of Florida's native and endemic wildlife species, including birds, mammals, reptiles, amphibians, insects, and marine life, as well as information about various plant species.

Feature 2: Detailed Species Profiles

- **How to Use:**
 - Log in to the app
 - Select from a list of species

- Select between the “Info” tab, “Cry” tab, and “Location” tab (Note: Location tab in development)
- **Benefits:**
 - Having detailed information can meet the app’s goal of being a comprehensive encyclopedia

Troubleshooting

- **Contact Support:**
 - Create a new issue on the GitHub repository with the issue, and at least one of our team members would reach out and help resolve it

Additional Tips

- This product is still in a demo version, so stay tuned for more features on the way

Thank You

Thank you for choosing FloriDex. We hope you enjoy using our app. If you have any questions or feedback, please don't hesitate to contact us.

Installation/Maintenance

1. Installation

1.1 System Requirements

- **Mobile Device:** Android device
- **Operating System:** Android 14 or higher

1.2 Installation Process

1. APK:

- Go to the GitHub repository <https://github.com/ColdCoder92/floridex> .
- Search for the APK floridex.apk.
- Tap the link to download the apk and install.
- Follow the on-screen instructions to complete the installation.

2. Maintenance

2.1 Automatic Updates

- Ensure your device is connected to Wi-Fi or mobile data and has the latest operating system version.

2.2 Manual Updates

- **Android:**
 - Go to the GitHub repository <https://github.com/ColdCoder92/floridex> .

- Search for the APK's latest version.
- Tap the link to download the apk and update.
- Follow the on-screen instructions to complete the installation.

3. Troubleshooting

- **App Crashes:**
 - Force-quit the app and restart it.
 - Update the app to the latest version.
 - Restart your device.
 - Contact our support team for further assistance.
- **App Won't Open:**
 - Ensure you have the latest operating system version.
 - Check your device's storage space.
 - Reinstall the app.
 - Contact our support team for further assistance.
- **App Not Loading Content:**
 - Check your internet connection.
 - Try restarting the app.
 - Clear the app's cache and data.
 - Contact our support team for further assistance.

4. Contact Support

- **Email:** coldcoder9225@protonmail.com
- **Website:** <https://github.com/ColdCoder92/floridex>

Shortcomings/Wishlist

Shortcomings

1. **Performance Issues:**
 - **Slow Load Times:** Description page experiences significant delay in loading
 - **Frequent App Crashes:** The app crashes unexpectedly under specific conditions or heavy usage.
 - **Limited Backward Compatibility:** The app's may not support devices below version 14
2. **User Interface/User Experience (UI/UX) Issues:**
 - **Limited Accessibility:** The app does not adequately support users with disabilities, such as color blindness or visual impairments.
3. **Feature Limitations:**
 - **Limited Customization Options:** Users have limited control over the app's appearance and behavior.

- **Insufficient Offline Functionality:** The app relies heavily on an internet connection, limiting its usability in areas with poor connectivity.
- 4. **Security and Privacy Concerns:**
 - **Weak Password Requirements:** The app's password requirements are not sufficiently stringent.
 - **Insufficient Data Encryption:** Sensitive user data may not be adequately encrypted during transmission and storage.
 - **Lack of Regular Security Updates:** The app may not receive timely security patches to address vulnerabilities.

Wishlist

1. **Performance Enhancements:**
 - Optimize app code and database queries for faster load times.
 - Implement efficient caching mechanisms to reduce network requests.
 - Improve app stability and reduce the frequency of crashes.
2. **UI/UX Improvements:**
 - Conduct usability testing to identify and address usability issues.
 - Redesign the app's navigation structure for better clarity and efficiency.
 - Implement accessibility features to support users with disabilities.
3. **Feature Expansion:**
 - Add new features to enhance the app's functionality and user experience including the ability to create creature entries and comments.
 - Provide more customization options to allow users to personalize the app and have a sense of identity like their profile picture.
 - Improve offline functionality by enabling key features to work without an internet connection.
4. **Security and Privacy:**
 - Strengthen password requirements to improve account security.
 - Implement robust encryption techniques to protect sensitive user data.
 - Regularly update the app with security patches to address vulnerabilities.

By addressing these shortcomings and implementing the desired features, we can significantly improve the user experience and overall quality of FloriDex.

REFERENCES

AWS. (2023). AWS Free Tier. Amazon Web Services, Inc. [https://aws.amazon.com/free/?all-free-tier.sort by=item.additionalFields.SortRank&all-free-tier.sort order=asc&awsf.Free%20Tier%20Types=](https://aws.amazon.com/free/?all-free-tier.sort%20by=item.additionalFields.SortRank&all-free-tier.sort%20order=asc&awsf.Free%20Tier%20Types=)

Developer Guides | Android Developers. (2019, January 23). Android Developers. <https://developer.android.com/guide/>

What is it · iNaturalist. (n.d.). INaturalist. <https://www.inaturalist.org/pages/what+is+it>