**System Design Document**

**For**

**ERAU Eagle Wallet**

Team Members:

* Samantha Balistreri
* Tyler Wise
* Joseph Moran
* Michael Fornito
* William Edwards
* Daniela Regueira

Table of Contents

[**1. Overview**](#_kzljllt38qf5) **1**

[**2. Introduction**](#_zhewvj9dksxg) **2**

[2.1 Purpose and Scope](#_1fob9te) 2

[2.2 Project Executive Summary](#_3dhxg8isrpxv) 2

[2.2.1 System Overview](#_7r1spg8hmp1d) 2

[2.2.2 Design Constraints](#_w6hyohxyqol2) 2

[2.2.3 Future Contingencies](#_auxj9ocjr48d) 2

[2.3 Glossary](#_zeeanxtnkv9c) 3

[**3. System Architecture**](#_yknai7s92qsu) **3**

[3.1 System Software Architecture](#_i96ojzvj0i8a) 3

[3.2 Internal Communications Architecture](#_xaq7586rkmwa) 4

[**4. Human-Machine Interface**](#_k6xsvhohs7o9) **4**

[4.1 Inputs](#_bwagjs7gap7e) 4

[4.1.1 Input Parameters](#_33z6ahyk9xyy) 4

[4.2 Outputs](#_z67nvlo70fqo) 4

[4.2.1 User Input Feedback](#_7f63pprw4lph) 5

[4.2.2 Warning Messages](#_6icih3pl2wdy) 5

[4.2.3 Data Collection](#_z49jyi7vxtj) 5

[**5. Detailed Design**](#_up9pyqenk91v) **6**

[5.1 Software Detailed Design](#_xl6j57ibkuz9) 6

[5.2 Overall System Design](#_i1lt7n1d4j9h) 7

[5.2.1 System Data Flow Diagram](#_ryujb1n7g0pl) 7

[5.3 API Design](#_sz792s4vvs11) 7

[5.3.1 API UML Class Diagram](#_2g2vwfw5nmx0) 7

[**6. External Interfaces**](#_do8lpsbmd8ew) **8**

[6.1 Interface Architecture](#_9bfbkeg6uz6s) 8

[6.2 Interface Detailed Design](#_cyymomd1vauq) 8

[**7. Additional Figures**](#_cg4pv6kginwp) **9**

[7.1 GUI For Human-Machine Interface](#_cl1petqgtiqv) 9

# 1. Overview

*The System Design Document describes the system requirements, operating environment, system and subsystem architecture, files and database design, input formats, output layouts, human-machine interfaces, detailed design, processing logic, and external interfaces.*

# 2. Introduction

## 2.1 Purpose and Scope

The purpose of this document is to show the architecture and design of the ERAU Eagle Wallet application which is constructed with Android Studio. The scope of this project is to develop an application that allows Embry-Riddle Aeronautical University faculty, staff, and students to access their Eagle Card and dining account through their phone. The application allows the user to make purchases through their phone using the near-field communication (NFC) reader.

## 2.2 Project Executive Summary

The ERAU Eagle Wallet application will be a useful application for Embry-Riddle faculty, staff, and students to manage their payment accounts and purchase food with touchless technology. This application will allow users to add funds to their account, check their current balances, pay for food, access the student dorms and access their Eagle Card.

## 2.2.1 System Overview

The product to be produced will be an application that will allow Embry-Riddle faculty, staff, and students to access their Eagle Card and dining account on their phone. This application will be required to interface with hardware in order to take advantage of the near-field communication.

## 2.2.2 Design Constraints

The system was designed to be used for Android 8.0 and above. Any user with an operating system below this will not be able to use the functionality of this app. This constraint is due to the implementation of biometrics and NFC capabilities available in the app.

## 2.2.3 Future Contingencies

One contingency that might arise that will affect the current design of the system is integrating the ERAU Eagle Wallet application with the official Embry-Riddle application. The second contingency is development for IOS devices using the apple developer licenses.

## 2.3 Glossary

* IDE - Integrated Development Environment
* Android Studio - IDE where the application will be constructed
* NFC - Near-Field Communication
  + A set of communication protocols for communication between two electronic devices over a distance of 4 cm (1​1⁄2 in)
* Android Versions
  + Android 8 - Oreo, minimum required android version for the application
  + Android 9 - Pie
* Emulator - hardware or software that permits programs written for one computer to be run on another computer.
* API - a software intermediary that allows two applications to talk to each other
* Widget - a component of an interface, that enables that enables a user to perform a function

# 3. System Architecture

## 3.1 System Software Architecture

In this section, describe the overall system software and organization. Include a list of software modules (this could include functions, subroutines, or classes), computer languages, and programming computer-aided software engineering tools (with a brief description of the function of each item). Use structured organization diagrams/object-oriented diagrams that show the various segmentation levels down to the lowest level. All features on the diagrams should have reference numbers and names. Include a narrative that expands on and enhances the understanding of the functional breakdown. If appropriate, use subsections to address each module.

All of the following software was used in development of this project:

* Android Studio
* Visual Studio
* Microsoft SQL Server Management Studio

The user application was built using Android Studio with Java support. Android studio supplied the means to design, create and add elements to the app. Java was used to add functionality to the elements implemented in the app and connected the backend database to the front end of the app.

The backend API and Database were built with C# and SQL, respectively. These two backend elements are stored on a dedicated server.

## 3.2 Internal Communications Architecture

In this section, describe the overall communications within the system; for example, LANs, buses, etc. Include the communications architecture(s) being implemented, such as X.25*,* Token Ring, etc. Provide a diagram depicting the communications path(s) between the system and subsystem modules. If appropriate, use subsections to address each architecture being employed.

**Note:** The diagrams should map to the FRD context diagrams.

# 4. Human-Machine Interface

This section provides the detailed design of the system and subsystem inputs and outputs relative to the user/operator.

## 4.1 Inputs

The figure in section 7 illustrates the graphical design of the application. This section describes input parameters for the Eagle Wallet Application based on the graphical design.

### 4.1.1 Input Parameters

1. **Username -** The user shall enter their username where prompted on the login page (This will be more than likely their school email address).
2. **Password -** The user shall enter their password where prompted on the login page
3. **Add Payment Information -** The user shall input their payment information onto their account for the ability to add funds. Payment information includes: card credentials, address, email, and name
4. **Account to Use** - The user shall click which account on the app they want to use for a purchase, this includes meal plans, dining dollars or eagle dollars.
5. **Remove Payment Information -** The user shall input the payment option that they choose to remove from their account
6. **Turn On NFC -** The user shall click the NFC icon to enable the NFC feature
7. **Turn Off NFC -** The user shall click the NFC icon to disable the NFC feature
8. **Add Funds -** The user shall add the amount of funds they want to their account (sodexo, dinning, and eagle dollars)
9. **Create Account -** The user shall create an account with the following parameters (username, password, email. studentID, full name)

## 4.2 Outputs

The figures in section 7 illustrates the graphical design of the application. This section describes output responses to ensure the user that all inputs are valid, notify the user with feedback, and to improve the software’s fault tolerance.

### 4.2.1 User Input Feedback

1. **Payment Added -** The app shall notify the user when a payment has been successfully added to their account
2. **Payment Deleted -** The app shall notify the user when a payment has been successfully deleted from the their account
3. **Purchase Completed -** The app shall notify the user when a purchase has gone through and been completed

### 4.2.2 Warning Messages

1. **Incorrect Username/Password -** This warning is displayed to the user if the user does not put the correct username or password into the login page.
2. **Payment method is not valid -** This warning is displayed to the user if the payment method they put into the app in order to add funds to their account is not a valid form of payment
3. **Insufficient Funds -** This warning is displayed to the user if the user makes a purchase and they do not have the necessary funds in their account
4. **Payment Failed -** This warning is displayed to user if the payment has failed due to something other than insufficient funds
5. **Required Fields -** The app shall notify the user when a particular field is left blank when trying to submit information.

### 4.2.3 Data Collection

1. **Eagle Wallet Database -** The database contains all the following information:

* Student name
* Email address
* Password
* ID Number
* Payment information
  + Meal Plans
  + Dining Dollars
  + Eagle Bucks

1. **Android System Information**

* User Biometrics
* System stored Google Pay information

# 5. Detailed Design

## 5.1 Software Detailed Design

A software module is the lowest level of design granularity in the system. Depending on the software development approach, there may be one or more modules per system. This section should provide enough detailed information about logic and data necessary to completely write source code for all modules in the system (and/or integrate COTS software programs).

If there are many modules or if the module documentation is extensive, place it in an appendix or reference a separate document. Add additional diagrams and information, if necessary, to describe each module, its functionality, and its hierarchy. Industry-standard module specification practices should be followed. Include the following information in the detailed module designs:

* A narrative description of each module, its function(s), the conditions under which it is used (called or scheduled for execution), its overall processing, logic, interfaces to other modules, interfaces to external systems, security requirements, etc.; explain any algorithms used by the module in detail
* For COTS packages, specify any call routines or bridging programs to integrate the package with the system and/or other COTS packages (for example, Dynamic Link Libraries)
* Data elements, record structures, and file structures associated with module input and output
* Graphical representation of the module processing, logic, flow of control, and algorithms, using an accepted diagramming approach (for example, structure charts, action diagrams, flowcharts, etc.)
* Data entry and data output graphics; define or reference associated data elements; if the project is large and complex or if the detailed module designs will be incorporated into a separate document, then it may be appropriate to repeat the screen information in this section
* Report layout

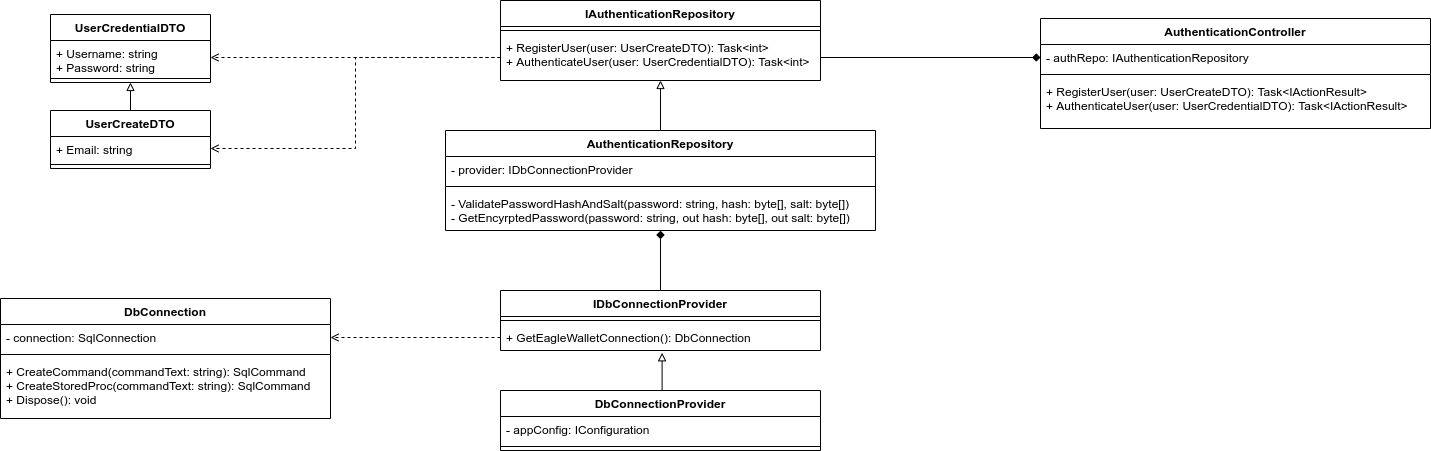
## 5.2 Overall System Design

## 5.2.1 System Data Flow Diagram

## 

## 5.3 API Design

## 5.3.1 API UML Class Diagram



# 6. External Interfaces

External systems are any systems that are not within the scope of the system under development, regardless whether the other systems are managed by the State or another agency. In this section, describe the electronic interface(s) between this system and each of the other systems and/or subsystem(s), emphasizing the point of view of the system being developed.

## 6.1 Interface Architecture

In this section, describe the interface(s) between the system being developed and other systems; for example, batch transfers, queries, etc. Include the interface architecture(s) being implemented, such as wide area networks, gateways, etc. Provide a diagram depicting the communications path(s) between this system and each of the other systems, which should map to the context diagrams in Section 1.2.1. If appropriate, use subsections to address each interface being implemented.

→ Google Pay

## 6.2 Interface Detailed Design

For each system that provides information exchange with the system under development, there is a requirement for rules governing the interface. This section should provide enough detailed information about the interface requirements to correctly format, transmit, and/or receive data across the interface. Include the following information in the detailed design for each interface (as appropriate):

* The data format requirements; if there is a need to reformat data before they are transmitted or after incoming data is received, tools and/or methods for the reformat process should be defined
* Specifications for hand-shaking protocols between the two systems; include the content and format of the information to be included in the hand-shake messages, the timing for exchanging these messages, and the steps to be taken when errors are identified
* Format(s) for error reports exchanged between the systems; should address the disposition of error reports; for example, retained in a file, sent to a printer, flag/alarm sent to the operator, etc.
* Graphical representation of the connectivity between systems, showing the direction of data flow
* Query and response descriptions

If a formal Interface Control Document (ICD) exists for a given interface, the information can be copied, or the ICD can be referenced in this section.

# 7. Additional Figures

This section illustrates additional figures regarding how the application is presented and functions for the user to ensure requirements are being met during the development process of the application.

## 7.1 GUI For Human-Machine Interface

