Final Progress Report

Team 4: Chandler Stevens, David Ang, Jason Djajasasmita, and Matthew Paik

GitHub Repository: <https://github.com/matt2849/CSC4151>

# Executive Summary

WalletWatch is an Android Application that helps people gain awareness of their spending habits. Our app aims to achieve this singular purpose through means of simplicity and ease of use. Our app gives users a focused view on the spending that matters the most to the user. The categories for spending can be set to whatever the user wants. Users simply open the app and log entries one at a time, giving it a name, and assigning it a category. Entries are automatically aggregated into a graph, which displays a line or pie chart on the analytics tab. This will help users quickly understand where and how they’ve been spending their money. If users make a mistake, they can open the history tab and edit or delete any expense. The data is encrypted locally on the user’s device as to ensure security. In addition to our goal of providing privacy for our users, our application allows users to setup a PIN as to restrict access to certain views. We desire for each of our users to have a no-frills experience that does not require any special permissions or personal information. Our app is offline, secure, and made for people who want a spending app that does exactly what it should, and nothing more.

# Target Audience

**Avatar:**



**Name:** James Gordon

**Age:** 26

**Location:** Seattle, WA

**Occupation:** IT Specialist

**Marital Status:** Single

**Annual Salary:** $70,000

**Reasons for needing our app:** James Gordon is a young professional that is trying to be more conscientious with his finances, as James currently lives in an apartment with a couple of roommates but wants to save up for a place to himself. He wants to try and spend less on groceries, subscriptions, and eating out to meet this goal, as well as keep his credit score high so he can request a house loan. He wants a secure, easy way to manage how much he’s spending on his Google Pixel 3. He is aware of apps like Mint but expresses concern at the risk involved with disclosing his banking information to a third-party company.

# Definition of Done

In order to consider our product to be “done” (have an MVP release candidate) our product needs to:

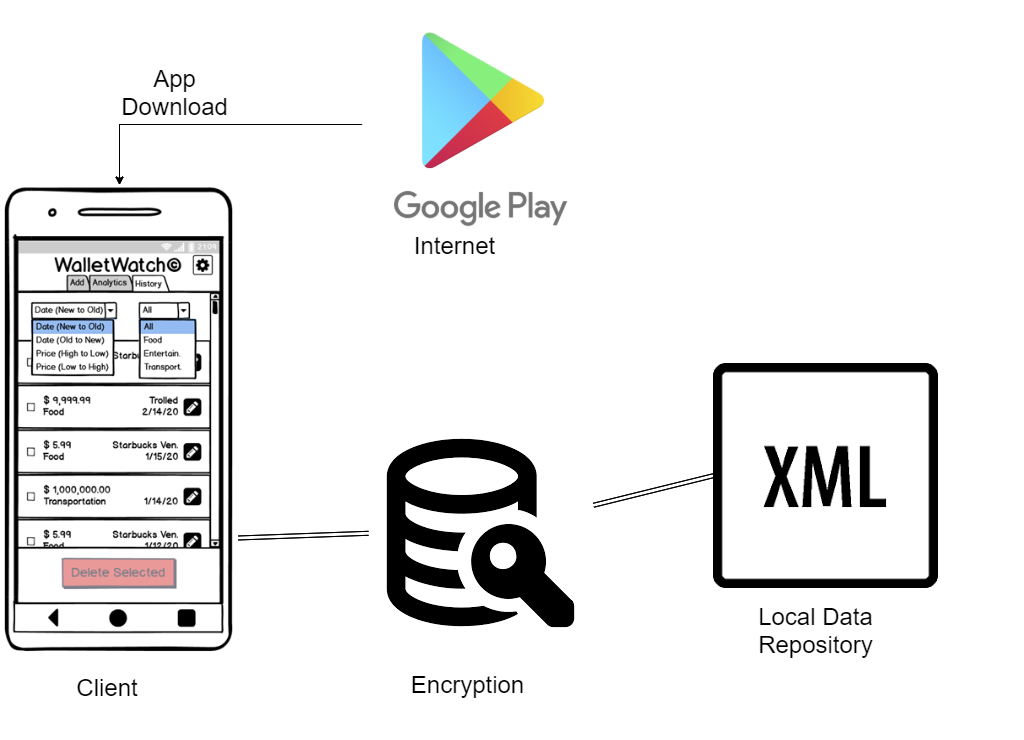
* Fulfill all the test plans including:
  + Unit Testing
  + Integration Testing
  + Function Testing
  + Acceptance Testing
  + Release Testing
  + \*Refer to [Test Repository.docx](https://teams.microsoft.com/l/file/C8ED9C81-202A-447C-AEE0-66D121A4075C?tenantId=d7270324-ea10-47a1-ae5f-74dba073f8fd&fileType=docx&objectUrl=https%3A%2F%2Fspuonline.sharepoint.com%2Fsites%2FCSC415xWS20202%2FShared%20Documents%2FTeam%204%2FTest%20Repository.docx&baseUrl=https%3A%2F%2Fspuonline.sharepoint.com%2Fsites%2FCSC415xWS20202&serviceName=teams&threadId=19:d636031255fd466f8aa323a89c54a6db@thread.skype&groupId=1128cb48-882f-4a74-b1f0-1db1209be26d) for more information\*
* Meet all the requirements of the Features List:
  + \*Refer to [Features List.docx](https://teams.microsoft.com/l/file/F4CC22BF-1D5E-4A16-818B-00C7471A326B?tenantId=d7270324-ea10-47a1-ae5f-74dba073f8fd&fileType=docx&objectUrl=https%3A%2F%2Fspuonline.sharepoint.com%2Fsites%2FCSC415xWS20202%2FShared%20Documents%2FTeam%204%2FFeatures%20List.docx&baseUrl=https%3A%2F%2Fspuonline.sharepoint.com%2Fsites%2FCSC415xWS20202&serviceName=teams&threadId=19:d636031255fd466f8aa323a89c54a6db@thread.skype&groupId=1128cb48-882f-4a74-b1f0-1db1209be26d) for more information\*

To meet these requirements, we will keep a schedule in our project management plan outlining all the things we must do on our backlog. Our process will be an Agile SCRUM methodology, which is a process in which every week we will be coding and holding team meetings, so that we will be able to track our progress. If we are falling behind, we can change our schedule and get back on track.

Our product will be done if and only if the test plan is satisfied and the integration of our feature list is completed. An incomplete test plan and/or feature list will mean that our product is not “done”.

System Architecture

The diagram below depicts an abstraction of our system architecture including the client application, the Kotlin-compatible encryption/decryption libraries, the local data repository, and the optional deployment to the Google Play Store:



# Feature List

To view the full Features List, view the following document: [Features List.docx](https://teams.microsoft.com/l/file/F4CC22BF-1D5E-4A16-818B-00C7471A326B?tenantId=d7270324-ea10-47a1-ae5f-74dba073f8fd&fileType=docx&objectUrl=https%3A%2F%2Fspuonline.sharepoint.com%2Fsites%2FCSC415xWS20202%2FShared%20Documents%2FTeam%204%2FFeatures%20List.docx&baseUrl=https%3A%2F%2Fspuonline.sharepoint.com%2Fsites%2FCSC415xWS20202&serviceName=teams&threadId=19:d636031255fd466f8aa323a89c54a6db@thread.skype&groupId=1128cb48-882f-4a74-b1f0-1db1209be26d)

To view the Functions List, view the following document: [Functions List.docx](https://teams.microsoft.com/l/file/5CDBE4FB-6EE5-4CF9-8C05-4C6EA4BC4F03?tenantId=d7270324-ea10-47a1-ae5f-74dba073f8fd&fileType=docx&objectUrl=https%3A%2F%2Fspuonline.sharepoint.com%2Fsites%2FCSC415xWS20202%2FShared%20Documents%2FTeam%204%2FFunctions%20List.docx&baseUrl=https%3A%2F%2Fspuonline.sharepoint.com%2Fsites%2FCSC415xWS20202&serviceName=teams&threadId=19:d636031255fd466f8aa323a89c54a6db@thread.skype&groupId=1128cb48-882f-4a74-b1f0-1db1209be26d)

To view the Constraints List, view the following document: [Constraints List.docx](https://teams.microsoft.com/l/file/E167DEDF-CDEE-4D84-9DBC-AC73DF40FCAF?tenantId=d7270324-ea10-47a1-ae5f-74dba073f8fd&fileType=docx&objectUrl=https%3A%2F%2Fspuonline.sharepoint.com%2Fsites%2FCSC415xWS20202%2FShared%20Documents%2FTeam%204%2FConstraints%20List.docx&baseUrl=https%3A%2F%2Fspuonline.sharepoint.com%2Fsites%2FCSC415xWS20202&serviceName=teams&threadId=19:d636031255fd466f8aa323a89c54a6db@thread.skype&groupId=1128cb48-882f-4a74-b1f0-1db1209be26d)

The following bullet-points provide a summarized version of the full features list:

1. Submitting New Entry (TAB 1: Default User View on App Launch)

* User must be able to create an entry without entering password (not password-protected)
* User must be able to specify dollar amount (USD) of expense
* User must be able to specify the category of expense
* Once user taps a category, the entry must immediately enqueue on to the verification queue
* Once user taps a category, a green temporary popup will notify of successful submission
* User could optionally specify the date of expense, which will be defaulted to the current date
* User could optionally specify identifier for expense, which will be defaulted to the current time

1. Viewing Analytics (TAB 2)

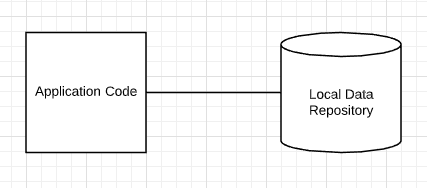
* This data must be PIN-protected upon viewing (in-app)
* User must be able to view line-graph of a subset of verified entries
* User must be able to view pie-chart of a subset of expense categories
* User must be able to select time interval (yearly, monthly, weekly, daily) for a subset

1. Viewing History (TAB 3)

* This data must be PIN-protected upon viewing (in-app)
* Users must be able to view a list of all previously submitted entries
* Each entry will have a price, date, and category and optional description
* The list should have scrolling functionality
* Each entry will have an edit button
* Each entry will have a checkbox
* At the bottom there will be a delete button
* The delete button will be initially greyed out, but upon selection of entries it will be enabled

# Internal Interfaces

The following diagram shows a basic view of the internal interfaces in which our application code connects to our encrypted XML local data repository



# External Interfaces

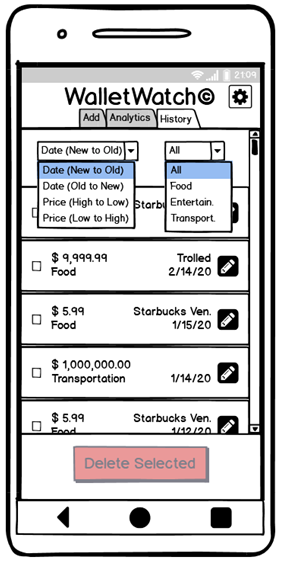
We do not anticipate any external interfaces connecting our system to any third-party services, especially since our application is completely offline.

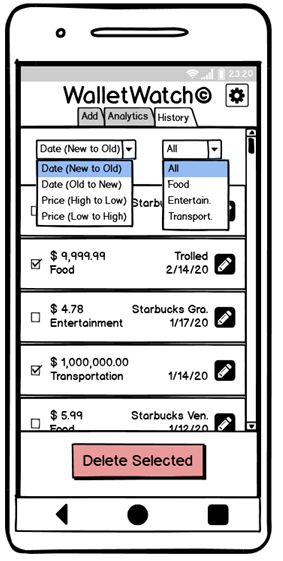
User Interfaces

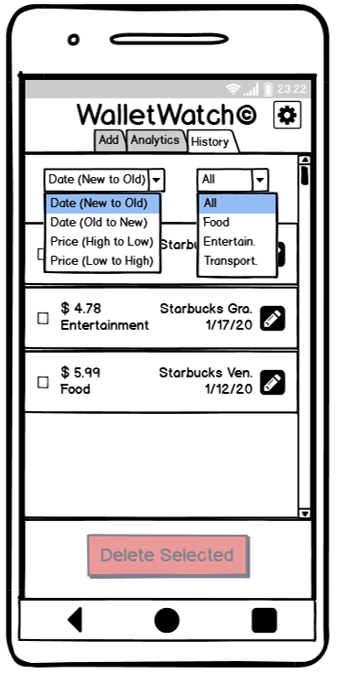
To view the low-fidelity prototype, view the following document: [WalletWatch Prototype.pdf](https://teams.microsoft.com/l/file/05EE4403-A85B-4C3D-BD1A-1841D1C10EA4?tenantId=d7270324-ea10-47a1-ae5f-74dba073f8fd&fileType=pdf&objectUrl=https%3A%2F%2Fspuonline.sharepoint.com%2Fsites%2FCSC415xWS20202%2FShared%20Documents%2FTeam%204%2FWalletWatch%20Prototype.pdf&baseUrl=https%3A%2F%2Fspuonline.sharepoint.com%2Fsites%2FCSC415xWS20202&serviceName=teams&threadId=19:d636031255fd466f8aa323a89c54a6db@thread.skype&groupId=1128cb48-882f-4a74-b1f0-1db1209be26d)

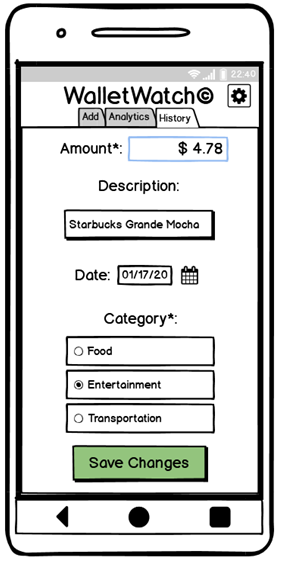
The following UI diagrams describe changes made to the UI after feedback was provided on the Second Progress Report Presentation. These changes completely removed the verification feature that was located on Tab 2. Instead, our team decided to move the content on Tab 3 (data analytics charts) over to Tab 2. We then decided to make Tab 3 a view for accessing, editing, and deleting expense history. This decision allowed our system to fully implement a CRUD methodology: Tab 1 allows users the ability to create expense data, Tab 2 allows users the ability to access aggregated expense data, Tab 3 allows users the ability to access, edit, and delete expense data, and the Settings tab allows users the ability to delete an entire category of expense data.

The following revised UI designs for Tab 3 will provide users the ability to view, edit, and delete data:

After receiving feedback in our previous progress report, we have decided to update our user interface to attempt to resolve the potential issues pointed out to us. We have changed the third tab from verification to history. We now have dropdown menus to help sort the history based on date or price as well as filter by category. Each entry displays a price, category name, date, and optional description. Each entry has a checkbox for deletion, and an edit button (pencil icon) on the right. By default, nothing is checked and the “Delete Selected” button is greyed out and disabled.

When the checkbox of an entry is selected, the “Delete Selected” button, which was previously greyed out becomes opaque to denote it is enabled, signifying that deletion is possible.

Upon hitting the “Delete Selected” button, the checked entries are deleted from the data repository and the list view as displayed here, and the “Delete Selected” button is once again greyed out to denote it is disabled.

When the user taps a pencil icon of one of the expenses, the expense editor will be displayed. This view is almost identical to the Tab 1: Add Entry screen displayed on app launch, with two key differences. Firstly, the category buttons have been transformed from pushbutton widgets into radio-button widgets. The reasoning behind this change is explained by the second difference, which is the “Save Changes” button that asks the user to confirm the changes and updates the local data repository. After tapping the “Save Changes” button or by canceling the edits by tapping the back arrow, the screen will display Tab 3: View History, which is provided in the first image in this User Interfaces section.

# Data Model

To view the full Data Model, view the following document: [Data Management.docx](https://teams.microsoft.com/l/file/48790EE5-0503-4E08-B59A-A7D675B0E2B7?tenantId=d7270324-ea10-47a1-ae5f-74dba073f8fd&fileType=docx&objectUrl=https%3A%2F%2Fspuonline.sharepoint.com%2Fsites%2FCSC415xWS20202%2FShared%20Documents%2FTeam%204%2FData%20Management.docx&baseUrl=https%3A%2F%2Fspuonline.sharepoint.com%2Fsites%2FCSC415xWS20202&serviceName=teams&threadId=19:d636031255fd466f8aa323a89c54a6db@thread.skype&groupId=1128cb48-882f-4a74-b1f0-1db1209be26d)

To view an example XML file, view the following document: [WalletWatch.xml](https://spuonline.sharepoint.com/sites/CSC415xWS20202/Shared%20Documents/Team%204/WalletWatch.xml)

Since our key distinctive feature from other financial tracking applications, such as Mint, is that our app will be a secure, offline mobile app, we will not have a relational database or any other form of online data repository. Instead, our app stores all data in an encrypted XML file that acts as a local data repository on the user’s Android device. Preliminary research led us toward various Kotlin-compatible libraries that will allow us to encrypt, decrypt, read, and write to the XML file utilizing Advanced Encryption Standard (AES) and Cipher Blocking Chain (CBC), which securely hash and salt the data.

# Project Management Plan

To view the full Task List, view the following document: [Task List.docx](https://teams.microsoft.com/l/file/C8ED9C81-202A-447C-AEE0-66D121A4075C?tenantId=d7270324-ea10-47a1-ae5f-74dba073f8fd&fileType=docx&objectUrl=https%3A%2F%2Fspuonline.sharepoint.com%2Fsites%2FCSC415xWS20202%2FShared%20Documents%2FTeam%204%2FTest%20Repository.docx&baseUrl=https%3A%2F%2Fspuonline.sharepoint.com%2Fsites%2FCSC415xWS20202&serviceName=teams&threadId=19:d636031255fd466f8aa323a89c54a6db@thread.skype&groupId=1128cb48-882f-4a74-b1f0-1db1209be26d)

We will utilize a SCRUM software development lifecycle methodology implemented by weekly Sprints, weekly virtual stand-up meetings, and utilization of Microsoft Teams to keep track of tasks that are on backlog, in progress, or completed. Chandler Stevens will act as SCRUM-master and will be responsible for facilitating virtual stand-up meetings.

The following bullet-points provide a summarized version of the full task list:

* Sprint 1 (3/22-3/28)
  + Setup Android Studio IDE, Kotlin, and GitKraken
* Sprint 2 (3/29-4/4)
  + Start implementing TAB 1: Add New Entry
* Sprint 3 (4/5-4-11)
  + Start implementing TAB 2: View Aggregated Analytics
* Sprint 4 (4/12-4/18)
  + Continue implementing TAB 2: View Aggregated Analytics
* Sprint 5 (4/19-4/25)
  + Continue implementing TAB 2: View Aggregated Analytics
* Sprint 6 (4/26-5/2)
  + Start implementing TAB 3: View Data History
* Sprint 7 (5/3-5/9)
  + Continue implementing TAB 3: View Data History
* Sprint 8 (5/10-5/16)
  + Start implementing settings button/window
* Sprint 9 (5/17-5/23)
  + Implement PIN verification window
* Sprint 10 (5/24-5/30)
  + Bug fixes and optionally add extra features

# Test Plan

To view the full Test Repository, view the following document: [Test Repository.docx](https://teams.microsoft.com/l/file/C8ED9C81-202A-447C-AEE0-66D121A4075C?tenantId=d7270324-ea10-47a1-ae5f-74dba073f8fd&fileType=docx&objectUrl=https%3A%2F%2Fspuonline.sharepoint.com%2Fsites%2FCSC415xWS20202%2FShared%20Documents%2FTeam%204%2FTest%20Repository.docx&baseUrl=https%3A%2F%2Fspuonline.sharepoint.com%2Fsites%2FCSC415xWS20202&serviceName=teams&threadId=19:d636031255fd466f8aa323a89c54a6db@thread.skype&groupId=1128cb48-882f-4a74-b1f0-1db1209be26d)

The following bullet-points provide a summarized version of the full test plan:

* Unit Tests (White-box: Functions perform correct actions and return correct outputs)
  + LaunchApp
  + SelectDate
  + SubmitEntry
  + ConfirmPIN
  + DisplayLineChart
  + SelectTimeInterval
  + SelectCategoryFilter
  + DisplayPieChart
  + SelectTimeType
  + SelectTimeValue
  + ChangeCategory
* Integration Tests (Bottom-up Approach: Modules work together correctly)
  + Tab 1
  + Tab 2
  + Tab 3
  + Settings
  + PIN Verification
* Functional Tests (Black-box: Features result in expected outcomes)
  + Tab 1
  + Tab 2
  + Tab 3
  + Settings
  + PIN Verification
* Acceptance Tests (User stories: Scenarios play out as expected)
  + Rapid Addition of an Expense without Past Date and Description
  + Addition of an Expense with Past Date and Description
  + View Various Analytics from Charts
  + View Expense History
  + Edit Previously Submitted Entry by Correcting Mistake
  + Delete Erroneous Entry
  + Edit Category to New Label
  + Opt-in for PIN and Setup Security Question
  + Utilize PIN Verification Feature
  + Determine Forgotten PIN
  + Change PIN
  + Opt-out of PIN
* Release Tests (Deployment: App can be downloaded and installed successfully)
  + Optionally deploy APK installer for completed MVP on the Google Play Store