Samuel:

Slide 1 - Title:

Good afternoon everyone, we are group 3 and today we will be presenting on our application, Tree Financial Tracker.

Slide 2 - Application Overview:

Tree Financial Tracker is a web-based financial tracking application. The application aims to aid users in being more frugal with their monetary expenses by tracking their financial habits. The budget calculator feature of the application also helps users ensure that they are spending within their limits in order to achieve their financial goals.

Tree Financial Tracker is a one-stop shop for users to collate all their financial information into a single application. Users can then add their daily transactions, ensuring that their financial information is always up to date. Users can also create budget goals for items that they want to purchase within a set period of time, and the application would recommend how much they should save per day/month and how much to set aside for other essentials, based on their historical transaction history.

Slide 3 - Expected Users:

Our application is mainly targeted at money-minded individuals that live in Singapore. Individuals who have an interest in saving and planning their finances for the future. Users who have difficulty keeping track of their finances, but are willing to follow the recommendation given by the system.

Slide 4 - Use Case Diagram:

This is our use case diagram

User will use: Register for a user account, Login, Add initial information, Track expenditure, Display summary on Dashboard, View Goals and Calculate Budget

Account System will use: User Account Creation, User Account Verification, Update Information, Reset Password.

Transaction System will use: Add new Transaction, Edit Transaction, Delete Transaction, View Transaction History

Goals System will use: Add Goals, Edit Goals, Delete Goals

Data.gov.sg API will use: Retrieve Relevant Datasets via API

Slide 5 - Good Practices:
Our application makes use of the Model View Controller (MVC) architectural pattern which allows for easy implementation for future functionality.

- Any changes to a particular section of the application would not affect the entire system.
- Adding/updating new views would also not affect the entire system

Slide 6 - Class Diagram:
Our application also makes use of OOP methodology, where the UI classes act as boundary classes. System and manager classes act as controllers that perform the logic and utility functions of the application, and finally, classes like the user, transaction, etc with persistent data act as entity classes. By segregating these classes into the following architectural pattern, we can then achieve the 5 principles of object-oriented design, SOLID. An example of a principle being applied would be the Single-Responsibility Principle. Each page in our application has its own UI class. Any changes made to one of the UIs would not affect the others.

Yao Xian:

Demo:
—-- Show new account creation —--

1. Registration
2. Forget Password
3. Login
4. Initial set-up and budget set-up
    a. Force new accounts to be set up, etc.

—-- Switch to a populated account —--

5. Dashboard
6. Account
7. Transaction
8. Budget
9. Interfacing with other systems (show list of items within budget table)

Video Only:

10. Exception Handling
    a. Registration
    b. Login
    c. Forget Password
    d. Add/Edit/Delete Transaction
    e. Add/Edit/Delete Budget

Functional Requirements

1. Registration
    1.1. The System must allow the user to register for a new user account.
        1.1.1. Information to include:
            1.1.1.1. Username
            1.1.1.2. First Name
            1.1.1.3. Last Name
            1.1.1.4. Valid Email Address
            1.1.1.5. Password & Confirm Password
        1.1.2. The system must validate that all required fields are filled up with valid values.
        1.1.3. The user account availability must be validated by the system.
    1.2. The System must be able to validate their user account.
        1.2.1. Information to include:
            1.2.1.1. Verification Code
        1.2.2. The system must validate that the required field is filled up.
        1.2.3. The system must validate that the verification code is correct.
        1.2.4. The system will display an error message if the verification code is invalid.

2. Login
    2.1. The user must be able to log in using their username and password.
    2.2. The system must validate that all required fields are filled up with valid values.
    2.3. The system must validate that the username and password are correct.
    2.4. The system will display an error message if the login information is invalid.
    2.5. The user must be able to change their password if they forgot their current password.
        2.5.1. The system must be able to email the user their verification code in order to reset their password.
    2.6. The system will redirect the user to their dashboard upon successful login.

3. Dashboard
    3.1. The system must allow new users to perform the initial set-up of their accounts.
        3.1.1. The user must be able to add new financial accounts.
            3.1.1.1. Information to include:
                3.1.1.1.1. Account Type
                3.1.1.1.2. Name of Account
                3.1.1.1.3. Value
    3.2. The system must allow the user to add more financial accounts after the initial set-up.
    3.3. The system must be able to display net worth as a graph.
    3.4. The system must be able to display the user's cash flow over time.
    3.5. The system must be able to display expenses by category in a pie chart.
    3.6. The system must be able to display income by category in a pie chart.

4. Account
    4.1. The system must be able to display the user's net worth, broken down into different financial accounts.
    4.2. The user must be able to add a new transaction and its relevant details.
        4.2.1. Details include:
            4.2.1.1. Transaction Type
            4.2.1.2. Transaction Account
            4.2.1.3. Date
            4.2.1.4. Category
            4.2.1.5. Transaction Name
            4.2.1.6. Amount
            4.2.1.7. Remarks
    4.3. The system must be able to display the monthly budget goal in the form of a progress bar.
    4.4. The system must be able to display the cash flow for the current month.
    4.5. The system must be able to display recent transactions made by the user.
    4.6. The system must be able to redirect the user to the transaction page to view all transactions.
    4.7. The user must be able to edit or delete their past transactions should the user choose to do so.

5. Budget
    5.1. The user must be able to set financial goals by entering their goal information.
        5.1.1. Information Includes:
            5.1.1.1. Priority
            5.1.1.2. Goal Name
            5.1.1.3. Value
            5.1.1.4. Target Duration
    5.2. The system must be able to display the estimated budget to the user
        5.2.1. The system must show how much money should be set aside by the user for essentials.
            5.2.1.1. Essentials Include:
                5.2.1.1.1. Food
                5.2.1.1.2. Housing
                5.2.1.1.3. Transportation
                5.2.1.1.4. Utilities
                5.2.1.1.5. Insurance
                5.2.1.1.6. Medical
                5.2.1.1.7. Personal
                5.2.1.1.8. Recreational
                5.2.1.1.9. Miscellaneous
        5.2.2. The system must display a suggested maximum amount the user can spend to attain their goals within the target duration.

5.2.2.1. The system must display the monthly progress and total progress towards the set goal.
5.2.2.2. The system must display whether the user is on track to attain their goals.
5.2.3. The system must be able to display the list of items that are within budget (e.g. Food, Dessert, Groceries, etc.) to the user.
5.2.3.1. Information Includes:
5.2.3.1.1. Item Name
5.2.3.1.2. Amount
5.2.3.2. The system must display a list of all items that are within the budget to the user via the table in the budget page.

6. Transaction
6.1. The system must be able to show the entire transaction details and history of the user.
6.1.1. Details include:
6.1.1.1. Date
6.1.1.2. Transaction Type
6.1.1.3. Transaction Account
6.1.1.4. Transaction Name
6.1.1.5. Remarks
6.1.1.6. Category
6.1.1.7. Amount
6.2. The user must be able to search by transaction name.
6.3. The user must be able to sort by transaction details.

7. Interfacing with other systems
7.1. The system must be able to retrieve the list of prices of common products and food items from Data.gov.sg API.

8. Data Format
8.1. Date must be in “DD/MM/YYYY” format.
8.2. Monetary values must be in “$” and two decimal places of accuracy.

Slide 8:
Thank you.