FlashIt

Fuller Computing

Gordon Huynh, Eric Le, Jimmy Xuan, Tommy Chao, Jens Bernardino, John Shelton

CPSC 362

Professor Ghadami

Table of Contents

Revision History

Date	Version	Description	Author(s)
9/14/2018	0.1	Created app framework using flutter.io within VSCode to begin development.	Jens Bernardino Gordon Huynh Eric Le
9/24/2018	0.1.1	Added toolbars, icon functionality, and user confirmation checks. Set proper paths on a screen to screen basis.	Jens Bernardino Gordon Huynh Eric Le Jimmy Xuan
10/15/2018	0.1.2	Implemented basic flashcard creation and local text-to-file saving with buttons, and other functionality such as reading and writing to text, as well as locating files within local directories for reading.	Jens Bernardino Gordon Huynh Eric Le Jimmy Xuan Tommy Chao
10/16/2018	0.1.3	Fixed flashcard creation and flashcard text-to-file saving. Removed unnecessary code and redundant functionalities.	Eric Le Tommy Chao Jimmy Xuan
10/16/2018	1.0	Implemented flashcard answer/question viewing, refactored code to be more encapsulated. Implemented cards visually.	Eric Le Jens Bernardino Jimmy Xuan Gordon Huynh Tommy Chao John Shelton

11/10/2018	1.1	Implemented basic deck system capable of loading several decks. Includes reading/writing errors; to be fixed later.	Jimmy Xuan Eric Le Gordon Huynh
11/13/2018	1.2	Implemented basic framework for test scoring and improved the quiz action. Added counter for checking number of times “Correct” was marked. Refactored code to be more encapsulated.	Tommy Chao
11/14/2018	1.3	Merged separate branches of code from different members, committed to master branch to allow development team to be working off the same branch.	Tommy Chao Eric Le Jimmy Xuan Gordon Huynh
11/16/2018	1.3.1	Implemented scoring system within the quiz feature when selecting a specific deck.	Tommy Chao Jens Bernardino
11/17/2018	1.3.2	Fixed small bug regarding page refreshing and improper screens.	John Shelton
11/17/2018	1.4	Implemented flashcard editing buttons for changing the questions and answers while viewing a flashcard.	Jens Bernardino
11/18/2018	2.0	Implemented deck logistics, tracks number of times test was attempted on a deck, and number of times a perfect score was achieved. Finalized second iteration of the application.	Tommy Chao
12/7/2018	2.1	Implemented sidebar for quick access to general settings and FAQs.	Tommy Chao
12/7/2018	2.1.1	Implemented brightness switching between a bright and dark theme over a toggle switch in the sidebar.	Gordon Huynh Jimmy Xuan
12/9/2018	3.0	Implemented contact feature that allows for feedback from app users to be received by developers.	Jimmy Xuan

Project Plan

As of December 1st, 2018, the development team initiated pre-game planning and elected our user stories based on their importance when further continuing development on our app. With many features of high importance having been implemented within the previous sprint, the development team chose to focus on quality-of-life changes and other features that would make the application more accessible. This included the implementation of a brightness setting and a sidebar on the home screen. Taking into consideration previous feedback of the client in which the client did not know what certain features performed, the development team implemented a FAQs page, containing directions for general use of the app.

The development team also chose to refine other features previously implemented to increase the accuracy of information and scores relayed back to the user. An example of this would be displaying a history of what previous test attempts were on a certain deck, allowing the user to have a better understanding of where their strengths and weaknesses of certain subjects lie.

Use-Cases (Textual)

Use Case Name: Create Flashcard	ID: UC-001
Primary Actor: App User	
Stakeholders and Interests: App Users, Clients	
Pre-Condition: User must be able to open the application on a phone.	
Post-Condition: User is able to create flashcard.	
Brief Description: User taps the appropriate button to navigate to the "Create Flashcard" tab. User enters a question and answer and taps "save". The question is then saved to a text file, and the flashcard is created.	
Main Success Scenario: 1. User taps the Create Flashcard button. 2. User enters a question in the text field, enters an answer in ht pop-up dialogue box. 3. User taps the save button, and creates a flashcard.	

Use Case Name: View All Flashcards	ID: UC-002
Primary Actor: App User	
Stakeholders and Interests: App Users, Clients	
Pre-Condition: User must be able to open the application on a phone.	
Post-Condition: User is able to view and manage flashcards.	
Brief Description: User taps the Manage Flashcards button to view all sets, flashcards and answers.	
Main Success Scenario: 1. User taps the Manage Flashcards button. 2. User is able to view all sets, flashcards, and answers.	

Use Case Name: Delete All Flashcards	ID: UC-003
Primary Actor: App User	
Stakeholders and Interests: App Users, Clients	
Pre-Condition: User must be able to open the application on a phone.	
Post-Condition: User is able to delete all flashcards.	
Brief Description: User taps the appropriate button to navigate to the "Manage Flashcards tab. User taps the "Delete Flashcards" button. All previous flashcards are deleted.	
Main Success Scenario: 1. User taps the Manage Flashcards button. 2. User taps the Delete Flashcards button.	

Use Case Name: Quiz	ID: UC-004
Primary Actor: App User	
Stakeholders and Interests: App Users, Clients	
Pre-Condition: User must be able to open the application on a phone.	
Post-Condition: User is able to view flashcard questions and view answers. User is able to mark whether or not they answered correctly.	
Brief Description: User taps the "Test Flashcards" button to test self. Questions appear at random and the user can tap the question box to show the answer. User can then tap the check to show that the question was answered correctly.	
Main Success Scenario: 1. User taps the Test Flashcards button. 2. A question will appear at random, user can tap the appropriate button to view the answer. 3. User is select whether or not they answered correctly or incorrectly.	

Use Case Name: Return Home/ Exit	ID: UC-005
Primary Actor: App User	
Stakeholders and Interests: App Users, Clients	
Pre-Condition: User must be able to open the application on a phone.	
Post-Condition: User is on the home screen and is able to exit the application.	
Brief Description: User is able to return to the home screen from any page and exit the application.	
Main Success Scenario: 1. User is on any screen and taps the home button. 2. On the home screen, user taps the "Exit" button. 3. The application closes.	

Use Case Name: Creating Deck	ID: UC-006
Primary Actor: App User	
Stakeholders and Interests: App Users, Clients	
Pre-Condition: User must be able to open the application on a phone.	
Post-Condition: User is able to create a deck of flashcards grouped by subject.	
Brief Description: User taps "Create Deck", and is then prompted to enter a name for the deck. Tapping the "Create" button will then create and empty deck, which the user can then add cards to.	
Main Success Scenario: 1. User taps the "Create Deck" button. 2. User then enters a name for the deck, and taps the "Create" button. 3. An empty deck will be created	

Use Case Name: Editing a Flashcard	ID: UC-007
Primary Actor: App User	
Stakeholders and Interests: App Users, Clients	
Pre-Condition: User must be able to open the application on a phone.	
Post-Condition: User is able to edit a flashcard created beforehand..	
Brief Description: Editing of previously created flashcards in order to avoid the need of creating a new flashcard from scratch.	
Main Success Scenario: 1. User taps “Manage Flashcards” button. 2. User navigates to the desired flashcard and taps the “Edit Flashcard” button on the bottom application bar. 3. User can then edit the question and answer at will.	

Use Case Name: Test Scoring/Results	ID: UC-008
Primary Actor: App User	
Stakeholders and Interests: App Users, Clients	
Pre-Condition: User must be able to open the application on a phone.	
Post-Condition: User is test oneself and view correct answers and overall score of the set.	
Brief Description: Testing feature of the app is improved through the implementation of a scoring system. App now tracks which questions were answered properly or improperly, and will be displayed once the flashcard set is perused.	
Main Success Scenario: 1. User taps “Test Flashcards” button. 2. User traverses through all cards within the set. 3. The results will be displayed once all cards within the set have been marked with “correct” or “incorrect”.	

Use Case Name: Recent Deck Access	ID: UC-009
Primary Actor: App User	
Stakeholders and Interests: App Users, Clients	
Pre-Condition: User must be able to open the application on a phone.	
Post-Condition: User is able to access recent decks created/viewed from the home page.	
Brief Description: Recent decks are accessible through the home screen through button presses, increasing the accessibility and convenience of studying.	
Main Success Scenario: 1. User accesses the home screen. 2. Deck names are displayed as separately; user can tap each respective deck name. 3. User can view all questions and answers of that deck.	

Use Case Name: View Deck Logistics	ID: UC-010
Primary Actor: App User	
Stakeholders and Interests: App Users, Clients	
Pre-Condition: User must be able to open the application on a phone.	
Post-Condition: User is able to view information pertaining to the deck..	
Brief Description: User can view info regarding their performance pertaining to each deck created, such as the number of times a test has been attempted on the deck, and the number of times a perfect score was achieved on the test.	
Main Success Scenario: 1. User taps the “Manage Flashcards” button. 2. User taps “My Sets” button. 3. User can press and hold a set to view its logistics.	

Use Case Name: Bug Reports	ID: UC-011
Primary Actor: App User	
Stakeholders and Interests: App User	
Pre-Condition: User must be able to open the application on a phone.	
Post-Condition: User is able to access the side bar, and send a bug report to the developers.	
Brief Description: User can send feedback and bug reports to the development team.	
Main Success Scenario: 1. User taps the “SideBar” button on the top left of the homescreen. 2. User taps “Contact Us” tile on the tile list. 3. User enters message to the developers in the provided text field. 4. User taps “Send”, and the message will be forward to the specified email address.	

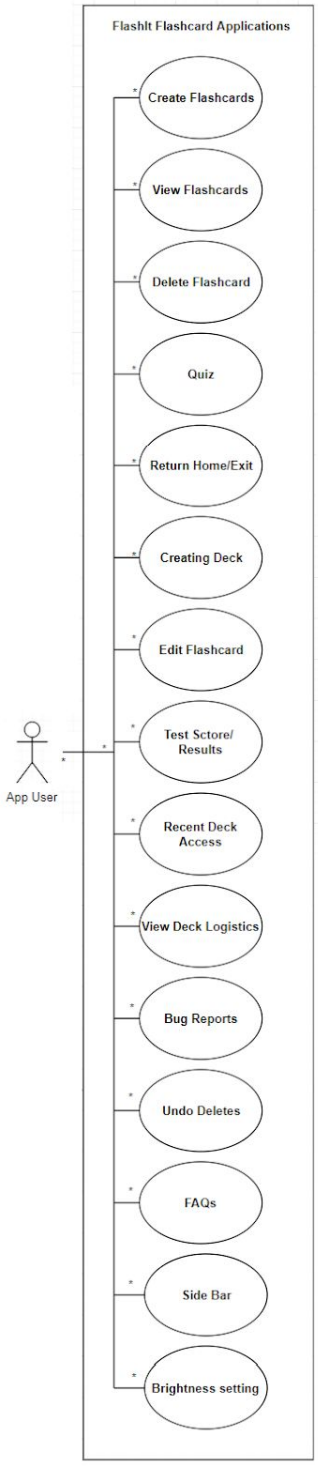
Use Case Name: Undo Delete	ID: UC-012
Primary Actor: App User	
Stakeholders and Interests: App User	
Pre-Condition: User must be able to access the application on a phone, and user must have deleted a flashcard or deck while browsing the “Manage Flashcards” page..	
Post-Condition: User is able to undo a previous delete within the “Manage Flashcards” page.	
Brief Description: After deleting a deck or flashcard, users can tap the “Swipe” icon in order to undo a previous delete.	
Main Success Scenario: 1. User selects a deck from the “Manage Flashcards” tab that has had a previous delete. 2. User taps the “Undo” button in the bottom app bar. 3. The previously deleted deck/flashcard is restored.	

Use Case Name: FAQs	ID: UC-013
Primary Actor: App User	
Stakeholders and Interests: App User	
Pre-Condition: User must be able to open the application on a phone.	
Post-Condition: User is able to view frequently asked questions.	
Brief Description: User can view frequently asked questions so they can have directions on how to use the app.	
Main Success Scenario: 1. User opens the application and is on the homescreen. 2. User taps the "SideBar" button on the top left of the screen. 3. User taps the FAQs tile on the side bar. 4. User is able to view frequently asked questions and their answers.	

Use Case Name: Side Bar	ID: UC-014
Primary Actor: App User	
Stakeholders and Interests: App User	
Pre-Condition: User must be able to open the application on a phone.	
Post-Condition: User is able to access the side bar, containing other functions.	
Brief Description: User can access the side bar, containing the brightness settings and other features.	
Main Success Scenario: 1. User opens the application and is brought to the homescreen. 2. User taps the "SideBar" button on the top left on the screen. 3. User is able to view features such as the brightness setting, FAQs, and bug reports.	

Use Case Name: Brightness Settings	ID: UC-015
Primary Actor: App User	
Stakeholders and Interests: App Users, Clients	
Pre-Condition: User must be able to open the application on a phone.	
Post-Condition: User is able to toggle a dark and light theme on the application.	
Brief Description: User is able to return to the home screen from any page and exit the application.	
Main Success Scenario: 1. User taps the “SideBar” button on the top left of the screen. 2. User slides the toggle to switch between dark and light themes.	

Use-Case Diagram



User Stories

I. As a user, I want to be able to create custom flashcards with self written questions and answers and be able to flip back and forth so that I can quickly memorize notes.

II. As a user, I want to be able to easily delete old flashcards so that I can clean out old flashcards that aren't used anymore in order to organize my flashcards.

III. As a user, I want to be able to view my flashcards together so that I don't have to flip between different pages to view each question.

IV. As a user, I want to be able to navigate between different screens of the application with a single button, so that I can quickly cycle through different sets.

V. As a user, I want to be able to test myself so that I can use the application as an effective study tool.

VI. As a user, I want to be able to create a deck of flashcards, so that I can group flashcards of the same subject together.

VII. As a user, I want to be able to edit an existing flashcard, so that I don't have to create a new flashcard from scratch because of a small mistake.

VIII. As a user, I want to be able to view my overall score after a quiz, so that I can gauge how well I know my study material.

IX. As a user, I want to be able to access my most recently accessed deck from the home screen, so that I can quickly resume studying if I choose to reopen the application after closing it.

X. As a user, I want to be able to view information such as how well I perform on quizzes of specific decks, so that I can accurately gauge what kind of material I need additional study in.

XI. As a user, I want to be able to send bug reports to the developers, so that I can give feedback quickly to the developers.

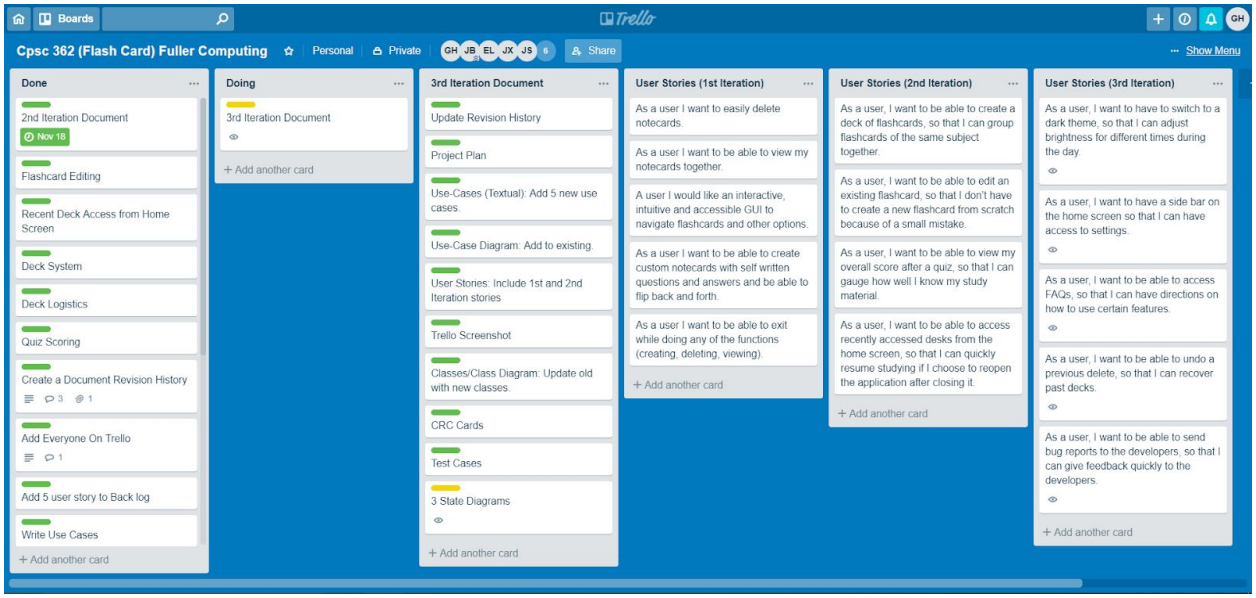
XII. As a user, I want to be able to undo a previous delete, so that I can recover past decks/flashcards.

XIII. As a user, I want to be able to access FAQs, so that I can have directions on how to use certain features.

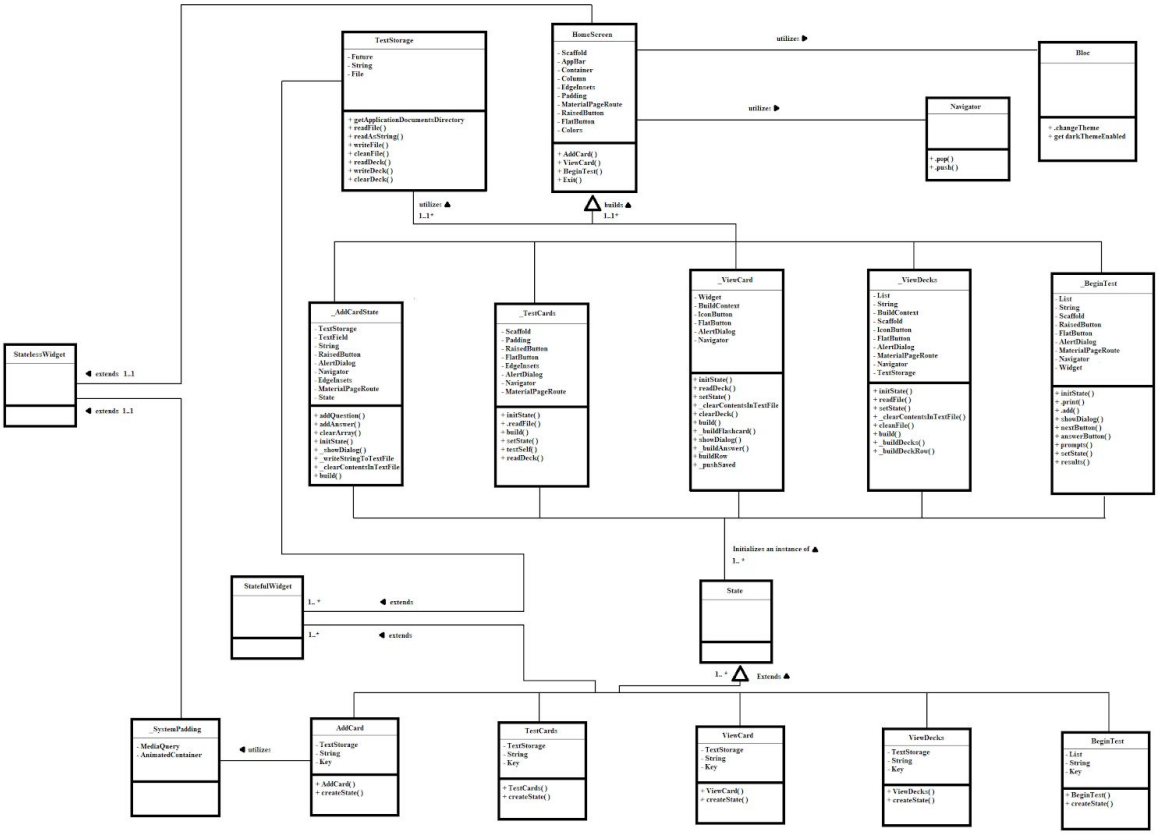
XIV. As a user, I want to have a sidebar on the home screen so that I can have access to settings.

XV. As a user, I want to have to switch to a dark theme, so that I can adjust brightness for different times during the day.

Trello Screenshot



Classes/Class Diagram



CRC Cards

Class Name: HomeScreen	ID: 01	Type: Concrete, Domain
Description: HomeScreen is responsible for handling the home page of the application. Initial page of the app that scaffolds other buttons to navigate to other pages.		Associated Use Cases: 10
Responsibilities - Contains buttons/widgets to access other pages. - Manages visuals and spacing of widgets on the home page.		Collaborators - StatelessWidget - Navigator - AddCard - ViewCard - ViewDecks - TestCards - TextStorage

Attributes:

- Scaffold
- AppBar
- Container
- Column
- EdgeInsets
- Padding
- MaterialPageRoute
- RaisedButton
- FlatButton
- Colors

Relationships:

- Generalization: N/A
- Aggregation: AddCard, ViewCard, ViewDecks, TestCard, TextStorage
- Other Associations: N/A

Class Name: AddCard	ID: 02	Type: Abstract
Description: AddCard contains necessary declaration of variables and initiates an instance of class _AddCardState.		Associated Use Cases: 3
Responsibilities - Declares necessary variables “storage” and “filename”. - Creates a mutable state of _AddCardState(), allowing it to be called several times over the lifetime of “StatefulWidget”.		Collaborators - StatefulWidget - _AddCardState

Attributes:

- TextStorage
- String
- Key

Relationships:

- Generalization: HomeScreen
- Aggregation: _AddCardState, _SystemPadding
- Other Associations: N/A

Class Name: _AddCardState	ID: 03	Type: Concrete, Domain

Description: _AddCardState includes functions to store user input within arrays. Declares respective text fields for user input retrieval. Respective arrays are stored to a .txt file for later retrieval. Creates screen overlay for the flashcard creation page.	Associated Use Cases: 3

Responsibilities	Collaborators
- Declares arrays _question and _answer, and accepts user input within respective input fields. - Declares question and answer text fields. - Accepted user input is written to a select text file for later access and retrieval during testing and viewing decks. - Create screen overlay for flashcard creation.	- State - StatefulWidget - AddCard

Attributes:

- TextStorage
- TextField
- String
- RaisedButton
- AlertDialog
- Navigator
- EdgeInsets
- MaterialPageRoute

Relationships:

- Generalization: AddCard
- Aggregation: AddCard, _SystemPadding
- Other Associations: N/A

Class Name: _SystemPadding	ID: 04	Type: Abstract, Domain
Description: _SystemPadding retrieves the size of the current overlay of the screen, and adjusts the positioning of widgets in the overlay.		Associated Use Cases: 10
Responsibilities - Calls “MediaQuery”, which gets the size of the insets of the current screen overlay. - Reutnrs a new “AnimatedContainer” of adjusted padding from the previous call of “MediaQuery”.		Collaborators - StatelessWidget

Attributes:

- MediaQuery
- AnimatedContainer

Relationships:

- Generalization: HomeScreen
- Aggregation: AddCard, _AddCardState
- Other Associations: N/A

Class Name: TestCards	ID: 05	Type: Abstract
Description: TestCards contains necessary declarations of variables and initiates an instance of the class _TestCards.		Associated Use Cases: 2
Responsibilities - Declares “storage” variable for accepting strings, which is then passed in as a parameter to the recursive call of TestCards. - Creates an instance of _TestCards.		Collaborators - StatefulWidget - _TestCards

Attributes:

- TextStorage
- Key

Relationships:

- Generalization: HomeScreen
- Aggregation: _TestCards
- Other Associations: N/A

Class Name: _TestCards	ID: 06	Type: Concrete, Domain
Description: Reads in data from .txt file, and generates appropriate pop up dialogue boxes and user interface for the quiz function. Questions and answers read in from the .txt file will be outputted.		Associated Use Cases: 2
Responsibilities - Reads in data from the .txt file, and outputs appropriate questions and answers during the pop up dialogue boxes during the quiz. - Generates the aforementioned pop up boxes and buttons for marking correct or incorrect answers. - Performs the quiz screen by screen.		Collaborators - State - StatefulWidget - TestCards

Attributes:

- Scaffold
- Padding
- RaisedButton
- FlatButton
- EdgeInsets
- AlertDialog
- Navigator
- MaterialPageRoute

Relationships:

- Generalization: HomeScreen
- Aggregation: TestCards
- Other Associations: N/A

Class Name: TextStorage	ID: 07	Type: Abstract, Domain
Description: Contains several getter functions for retrieving the location of the directory that contains the .txt file where user input is saved, as well as functions for writing to the .txt file.		Associated Use Cases: 8
Responsibilities - Contains functions responsible for retrieving user input stored in a local .txt file. - Contains functions responsible for writing user input to the aforementioned local .txt file.		Collaborators - Future - String - File

Attributes:

- Future
- String
- File

Relationships:

- Generalization: HomeScreen
- Aggregation: ViewCard, _ViewCard, ViewDecks, _ViewDecks, AddCard, _AddCardState
- Other Associations: N/A

Class Name: ViewCard	ID: 08	Type: Abstract
Description: Contains declarations for necessary data retrieval from the .txt file. Creates an instance of the class _ViewCard.		Associated Use Cases: 4
Responsibilities - Declares variables “storage” and “filename” necessary for data retrieval, passes in variables to the recursive call of ViewCard. - Initiate an instance of the class _ViewCards.		Collaborators - _ViewCard - StatefulWidget

Attributes:

- TextStorage
- String
- Key

Relationships:

- Generalization: HomeScreen
- Aggregation: _ViewCards
- Other Associations: N/A

Class Name: _ViewCard	ID: 09	Type: Concrete, Domain
Description: Creates an instance of a set of type "String", and sets an appropriate "FontStyle", and retrieves user input from the .txt file. Contains necessary functions to display questions and answers on scaffolded widgets		Associated Use Cases: 4
Responsibilities - Declares sets of appropriate types to store retrieved user input from the .txt file. - Contains and calls appropriate functions to display selected questions and answers in the screen overlay when appropriate buttons are tapped		Collaborators - ViewCard - State

Attributes:

- Widget
- BuildContext
- IconButton
- FlatButton
- AlertDialog
- Navigator

Relationships:

- Generalization: HomeScreen
- Aggregation: ViewCard
- Other Associations: N/A

Class Name: ViewDecks	ID: 10	Type: Abstract
Description: Contains necessary declarations for storage and initiates an instance of the class _ViewDecks.		Associated Use Cases: 5
Responsibilities - Declares the “storage” variable, which is then passed in as a parameter to the recursive call of ViewDecks		Collaborators - StatefulWidget - _ViewDecks

Attributes:

- TextStorage
- String
- Key

Relationships:

- Generalization: HomeScreen
- Aggregation: _ViewDecks
- Other Associations: N/A

Class Name: _ViewDecks	ID: 11	Type: Concrete, Domain
Description: Reads in user input from the .txt file, contains functions to create the screen overlay and view cards within the deck. Contains function to create an empty deck.		Associated Use Cases: 5
Responsibilities - Retrieve user input from the .txt file based on the appropriate button tapped when selecting a deck. - Build screen overlay and buttons to view flashcard questions and answers. - Contains functions regarding deck creation and adding flashcards to the said deck.		Collaborators - ViewDecks - ViewCard - State

Attributes:

- List
- String
- BuildContext
- Scaffold
- IconButton
- FlatButton
- AlertDialog
- MaterialPageRoute
- Navigator
- TextStorage

Relationships:

- Generalization: HomeScreen
- Aggregation: ViewDecks
- Other Associations: N/A

Class Name: BeginTest	ID: 12	Type: Abstract
Description: BeginTest contains necessary declarations of variables and initiates an instance of the class _BeginTest.		Associated Use Cases: 2
Responsibilities - Declares “deck” and “storage” variable for accepting strings, which is then passed in as a parameter to the recursive call of BeginTest. - Creates an instance of _BeginTest.		Collaborators - StatefulWidget - _BeginTest

Attributes:

- List
- String
- Key

Relationships:

- Generalization: HomeScreen
- Aggregation: _BeginTest
- Other Associations: N/A

Class Name: _BeginTest	ID: 13	Type: Concrete, Domain
Description: Contains iterator variables for keeping track of correct marks, and arrays for storing user inputted questions and answers retrieved from the .txt file. Contains functions for generating test overlay and paths.		Associated Use Cases: 2

Responsibilities	Collaborators
- Initialize iterator variables for number of correct marks. - Initialize arrays for storing questions and answers retrieved from the txt file. - Declares functions used during the test. - Generates the screen overlay with appropriate buttons and pop ups during the test function.	- BeginTest - State

Attributes:

- List
- String
- Scaffold
- RaisedButton
- FlatButton
- AlertDialog
- MaterialPageRoute
- Navigator
- Widget

Relationships:

- Generalization: HomeScreen
- Aggregation: BeginTest
- Other Associations: N/A

Class Name: Bloc	ID: 14	Type: Domain
Description: Initializes necessary variables for passing the stream for the dark theme toggle.		Associated Use Cases: 1
Responsibilities - Initializes variables such as "get darkThemeEnabled" and passes it to _themeController => stream. - Allows for dynamic theme changing across all screens of the app.		Collaborators - changedTheme( ) - darkThemeEnabled( )

Attributes: - final - stream
Relationships: - Generalization: HomeScreen - Aggregation: AddCard, BeginTest, TestView, Main - Other Associations: N/A

Test Cases

Test Case #: 1.0 **System**: Android **Designed by**: Fuller Computing **Executed by**: Fuller Computing **Short Description**: Test flashcard creation.	**Test Case Name**: Flashcard Creation **Subsystem**: Marshmallow 6.0 **Design Date**: 10/18/2018 **Execution Date**: 12/10/2018

Pre- Conditions

- User can access the application on an android machine.
- The application is on the homescreen.

Step	Action	Expected System Response	Pass/Fail	Comment
1	Tap the “Create New Deck” button.	The system will prompt the user to enter a deck name.		
2	Enter desired deck name.	The system will prompt the user to enter a question.		
3	Enter a question.	The system will pop up a text field to enter an answer.		
4	Enter an answer.	The system will prompt the user to save the card.		
5	Tap the “Save” button.	The system will then save the flashcard.		
6	Check Post-Condition 1.			

Post-Condition

- User has created a flashcard.

Test Case #: 2.0 **System**: Android **Designed by**: Fuller Computing **Executed by**: Fuller Computing **Short Description**: Test flashcard Viewing	**Test Case Name**: Flashcard Viewing **Subsystem**: Marshmallow 6.0 **Design Date**: 10/18/2018 **Execution Date**: 12/10/2018

Pre- Conditions

- User can access the application on an android machine.
- The application is on the homescreen.

Step	Action	Expected System Response	Pass/Fail	Comment
1	Click the "Manage Flashcards" button.	The system will display all your decks.		
2	Click the desired deck .	The system will display all the flashcards in the deck.		
3	Check Post-Condition 1			

Post-Condition

- User can view flashcards.

Test Case #: 3.0 **System**: Android **Designed by**: Fuller Computing **Executed by**: Fuller Computing **Short Description**: Test deck creation.	**Test Case Name**: Deck Creation **Subsystem**: Marshmallow 6.0 **Design Date**: 11/10/2018 **Execution Date**: 12/10/2018

Pre- Conditions

- User can access the application on an android machine.
- The application is on the homescreen.

Step	Action	Expected System Response	Pass/Fail	Comment
1	Tap the "Create New Deck" button.	The system will prompt the user to enter a deck name.		
2	Enter desired deck name.	The system will prompt the user to enter a question.		
3	Check Post-Condition 1.			

Post-Condition

- User has created a deck.

Test Case #: 4.0 **System**: Android **Designed by**: Fuller Computing **Executed by**: Fuller Computing **Short Description**: Test deck quizzing.	**Test Case Name**: Deck Quiz **Subsystem**: Marshmallow 6.0 **Design Date**: 11/10/2018 **Execution Date**: 12/10/2018

Pre- Conditions

- User can access the application on an android machine.
- The application is on the homescreen.

Step	Action	Expected System Response	Pass/Fail	Comment
1	Tap the “Test Flashcards” button.	The system will prompt the user to select a deck.		
2	Tap the desired deck to quiz.	The system will load in a question from the deck, and prompt the user to enter the answer within the provided text field.		
3	Enter answer, and tap the “Checkmark” icon..	The system will display the answer, and the user input next to each other, and mark whether or not the user answered correctly.		
4	Tap “OK” to continue.	The system will repeat the previous step with a different question.		
5	Continue to enter answers in the text fields.	The system will display your total score and the number of questions you answered correctly when all questions have been answered.		
6	Check Post-Condition 1.			

Post-Condition

- User is able to quiz a selected deck.

Test Case #: 5.0 **System**: Android **Designed by**: Fuller Computing **Executed by**: Fuller Computing **Short Description**: Deletes Flash Cards.	**Test Case Name**: Delete All Flash Cards **Subsystem**: Marshmallow 6.0 **Design Date**: 10/18/2018 **Execution Date**: 12/10/2018

Pre- Conditions

- User can access the application on an android machine.
- The application is on the homescreen.

Step	Action	Expected System Response	Pass/Fail	Comment
1	Tap the “Manage Flashcards” button.	The system will prompt the user to select a deck.		
2	Tap the Delete Flashcards” button	All previous flashcards are deleted.		
3	Check Post-Condition 1.			

Post-Condition

- User has deleted all flashcards

Test Case #: 6.0 **System**: Android **Designed by**: Fuller Computing **Executed by**: Fuller Computing **Short Description**: User is able to return to the Home Screen or exit the app.	**Test Case Name**:Return Home/Exit **Subsystem**: Marshmallow 6.0 **Design Date**: 10/18/2018 **Execution Date**: 12/10/2018

Pre- Conditions

- User can access the application on an android machine.
- User is on any page of the application

Step	Action	Expected System Response	Pass/Fail	Comment
1	Tap the “Home” button.	The system will take the user to the home screen.		
2	Tap the “Exit” button	The system closes the application.		
3	Check Post-Condition 1.			

Post-Condition

- User has successfully exited the application.

Test Case #: 7.0 **System**: Android **Designed by**: Fuller Computing **Executed by**: Fuller Computing **Short Description**: User can access the side bar.	**Test Case Name**: SideBar Access **Subsystem**: Marshmallow 6.0 **Design Date**: 10/18/2018 **Execution Date**: 12/10/2018

Pre- Conditions

- User can access the application on an android machine.
- The application is on the homescreen.

Step	Action	Expected System Response	Pass/Fail	Comment
1	Tap the icon in the top left of the homescreen.	The system will slide out a side bar containing FAQs, Dark Theme toggle, and Contact Us.		
2	Check Post-Condition 1.			

Post-Condition

- User can access the side bar.

Test Case #: 8.0 **System**: Android **Designed by**: Fuller Computing **Executed by**: Fuller Computing **Short Description**: Test flashcard editing	**Test Case Name**: Edit Flashcards **Subsystem**: Marshmallow 6.0 **Design Date**: 10/18/2018 **Execution Date**: 12/10/2018

Pre- Conditions

- User can access the application on an android machine.
- The application is on the homescreen.

Step	Action	Expected System Response	Pass/Fail	Comment
1	Tap the "Manage Flashcards" button.	The system will prompt the user to select a deck.		
2	Tap the desired flashcard to edit.	The system will display the flashcard.		
3	Click "Edit Question" or "Edit Answer" button.	The system will pop-up text field to enter new question or new answer. (Depending on which one you pick)		
4	Enter new question or answer and click enter.	The system updates the question or answer.		
5	Check Post-Condition 1.			

Post-Condition

- User has edited flashcard.

Test Case #: 9.0 **System**: Android **Designed by**: Fuller Computing **Executed by**: Fuller Computing **Short Description**: User is able to view FAQs	**Test Case Name**: View FAQs **Subsystem**: Marshmallow 6.0 **Design Date**: 12/07/2018 **Execution Date**: 12/10/2018

Pre- Conditions

- User can access the application on an android machine.
- User is on the homescreen.

Step	Action	Expected System Response	Pass/Fail	Comment
1	Tap the icon on the top left of the homescreen	The system will slide out a side bar containing FAQs, Dark Theme toggle, and Contact Us.		
2	Tap the “FAQs” tile.	The system displays several commonly asked questions.		
3	Tap a question to view the answer.	The system displays the answer to the specified FAQ.		
4	Check Post-Condition 1.			

Post-Condition

- User is able to view FAQs.

Test Case #: 10.0 **System**: Android **Designed by**: Fuller Computing **Executed by**: Fuller Computing **Short Description**: User is able contact the developers	**Test Case Name**: Contact Us **Subsystem**: Marshmallow 6.0 **Design Date**: 12/10/2018 **Execution Date**: 12/10/2018

Pre- Conditions

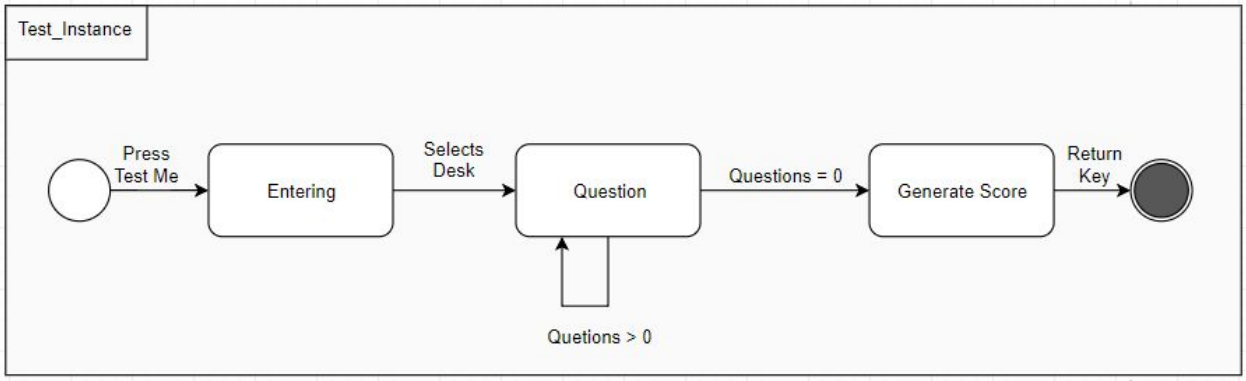
- User can access the application on an android machine.
- User is on the homescreen

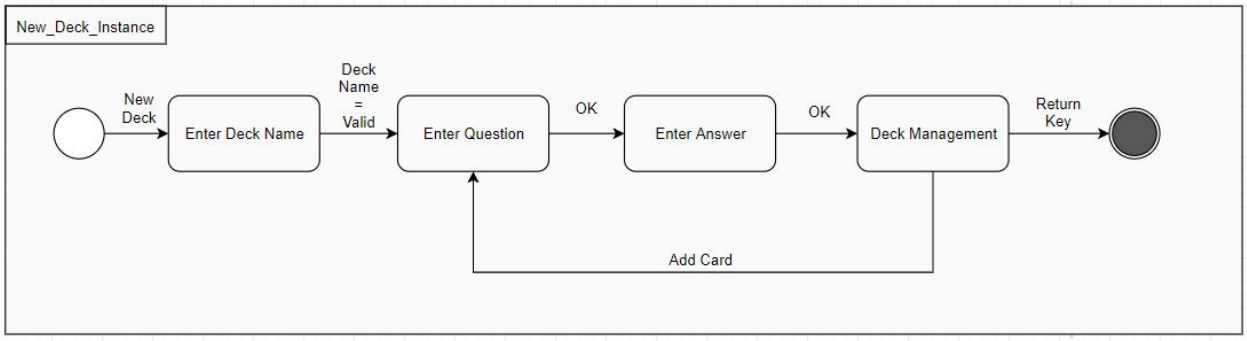
Step	Action	Expected System Response	Pass/Fail	Comment
1	Tap the icon on the top left of the homescreen	The system will slide out a side bar containing FAQs, Dark Theme toggle, and Contact Us.		
2	Tap the “Contact Us” button.	The system prompts the user to enter a message		
4	Tap the “Send” button.	The system sends the message to the developers		
3	Check Post-Condition 1.			

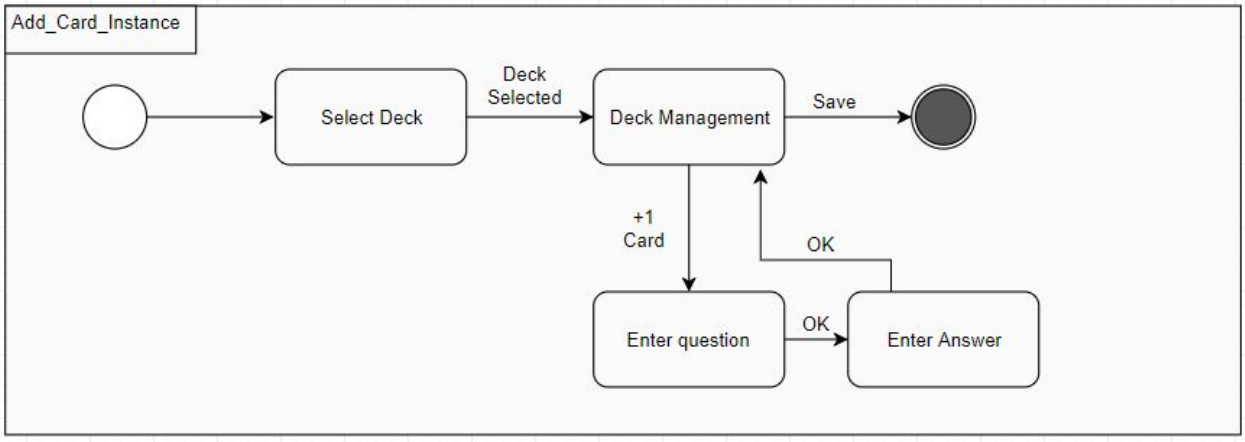
Post-Condition

- User has successfully sent a message to the developers.

State Diagrams



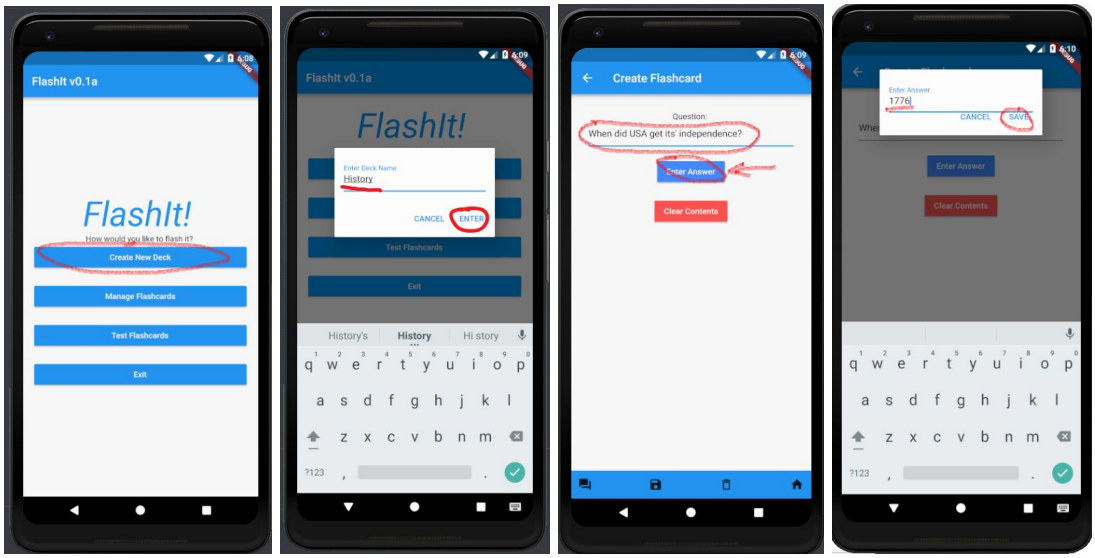


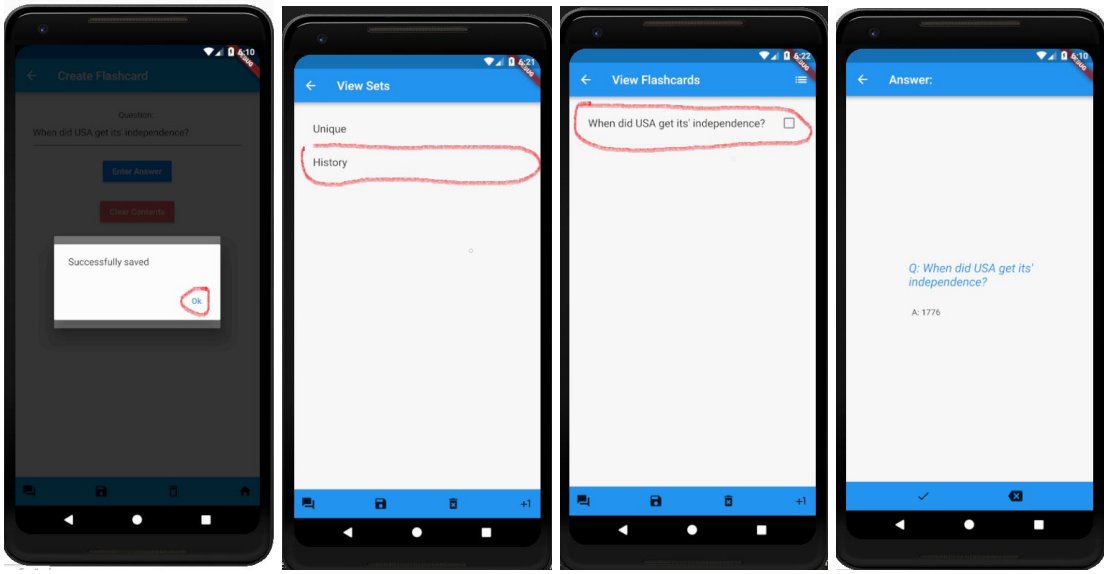


User Manual

1. Flashcard Creation

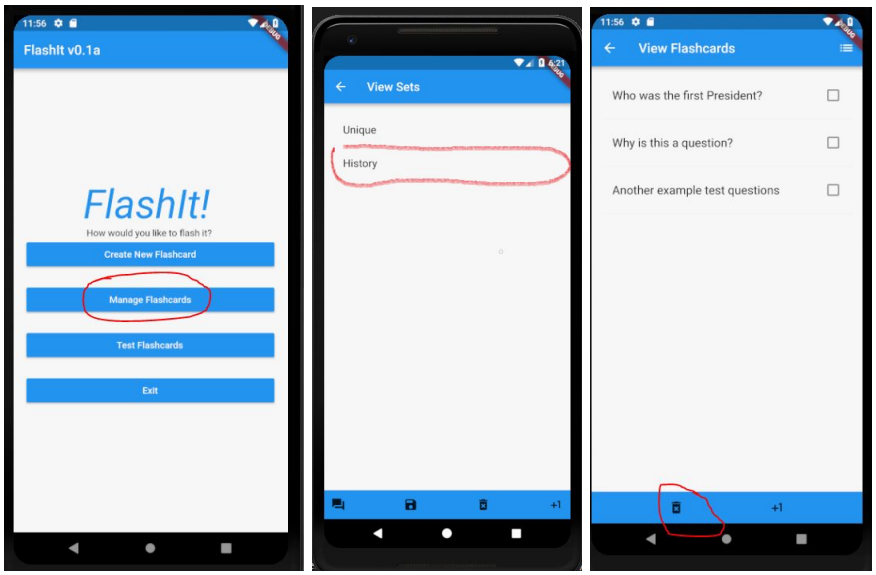
   User taps “Create New Deck” and enters a name for the deck. User is then prompted to enter a question and an answer. User taps “Save”, and the flashcard is created.

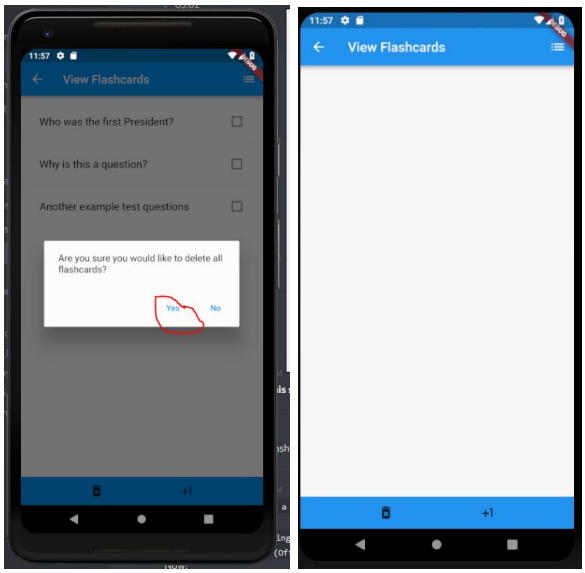




2. Flashcard Deletion

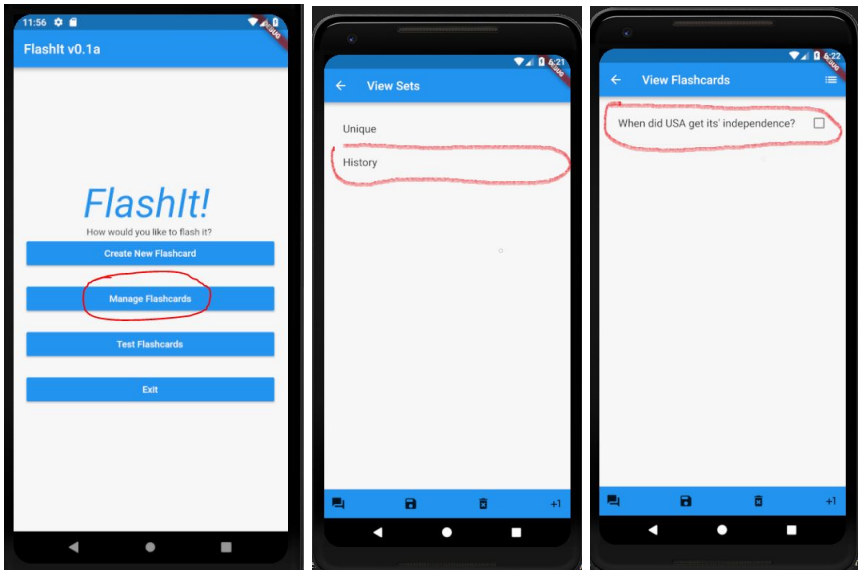
- Users may choose to delete their flashcards by tapping “Manage Flashcards”. Then, by tapping the recycle bin icon in the icon bar on the bottom, users will be prompted with a confirmation to delete their cards.





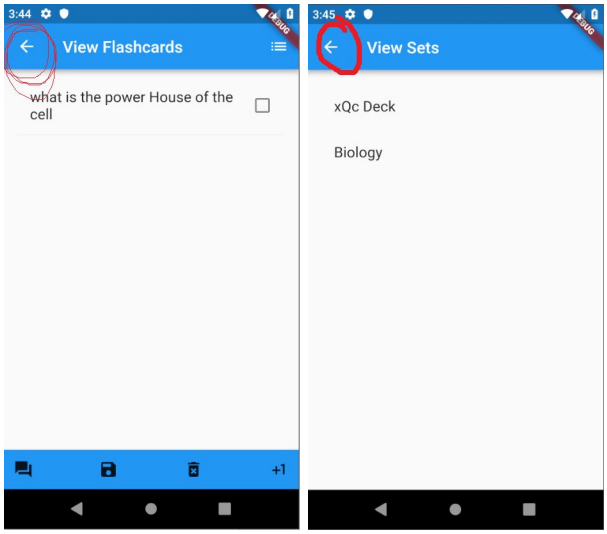
3. Viewing Flashcards

- Users may view their flashcards by tapping the “Manage Flashcards” app, selecting a deck, and viewing all flashcards within that deck.



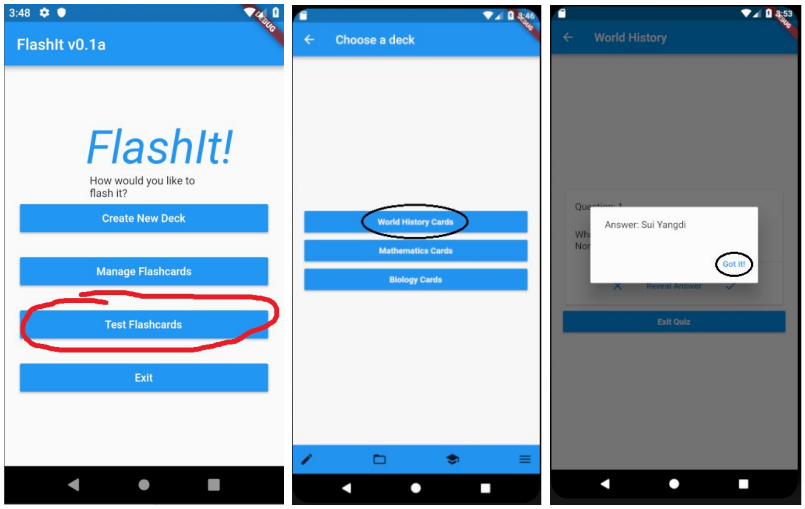
4. Fast Screen Navigation

- Users may quickly cycle to previous pages by tapping the “Back” button.



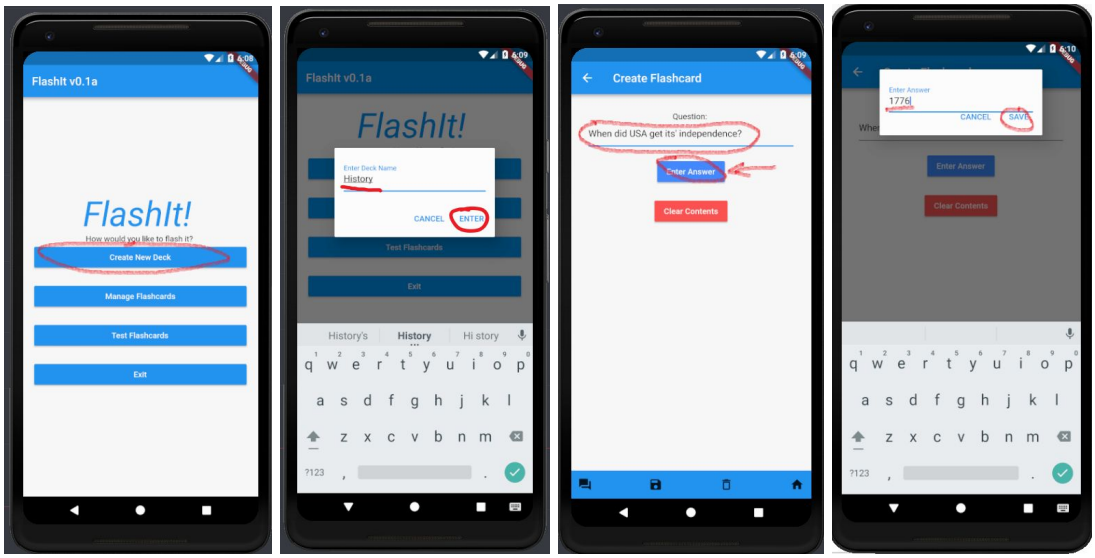
5. Test

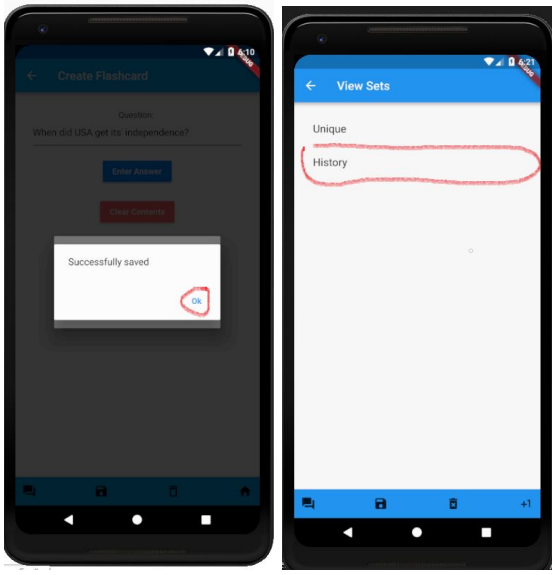
- Users may test themselves on a specific deck by tapping the “Test Flashcards” button the homescreen. Users can then select a deck, and questions will be shown on a new screen. Users may mark whether or not they answered correctly, and may reveal the answer if they choose.



6. Deck Creation

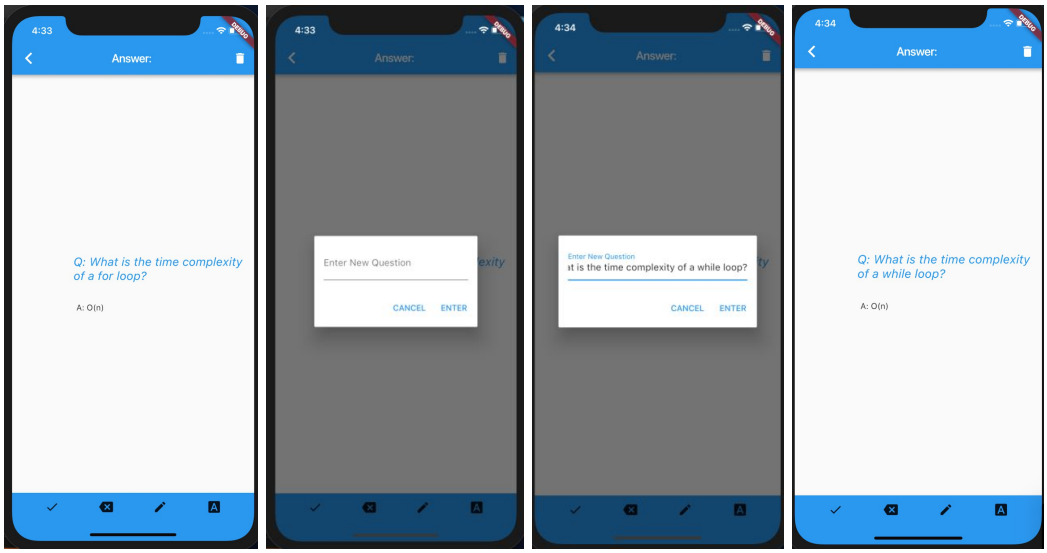
- User taps “Create New Deck” and enters a name for the deck. User is then prompted to enter a question and an answer. User taps “Save”, and the deck is created.

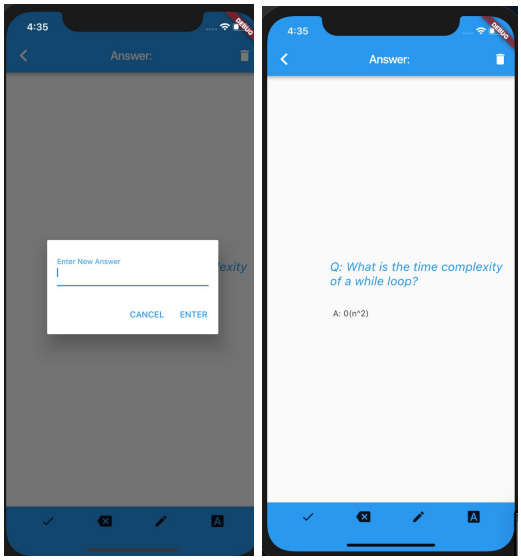




7. Flashcard Editing

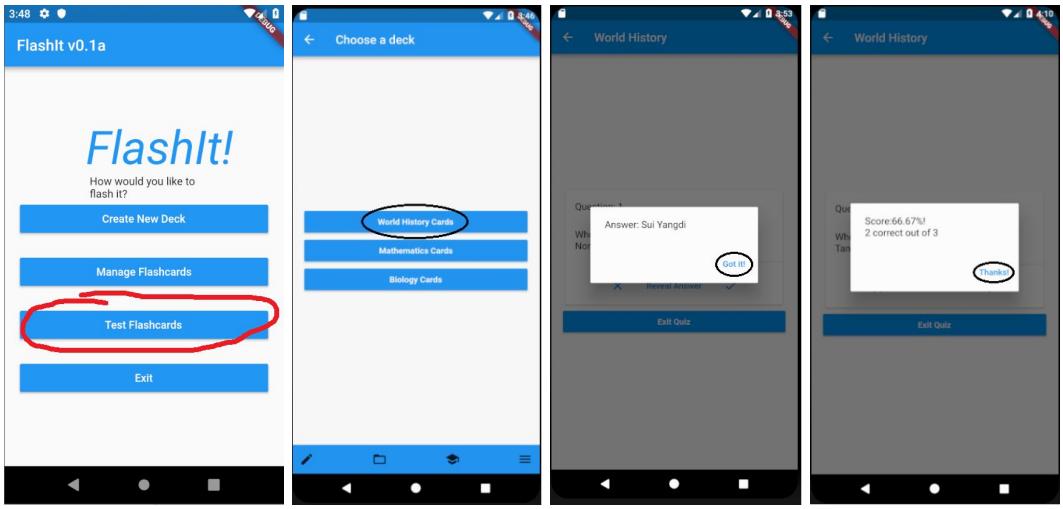
- When viewing a flashcard, users may tap the “Pencil” button on the bottom bar to enter a new question, or the “Answer” button on the bottom bar to enter a new answer. The new input will overwrite the previous question/answer in the .txt file.





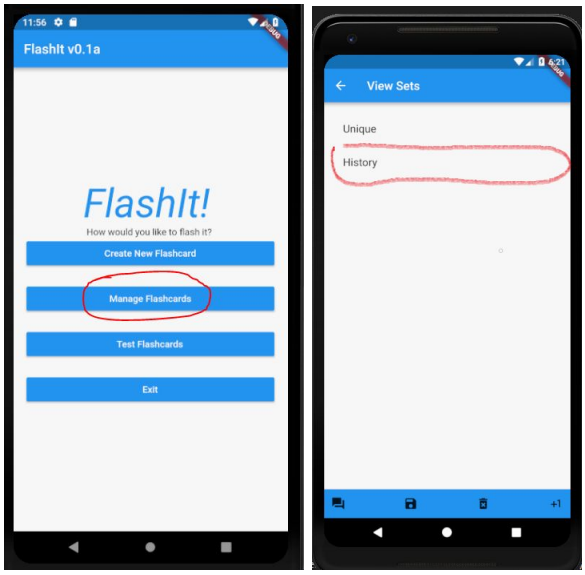
8. Test Scoring

- After a test, user may view their score after answering all questions of a certain set. The results are displayed automatically after answering all questions.



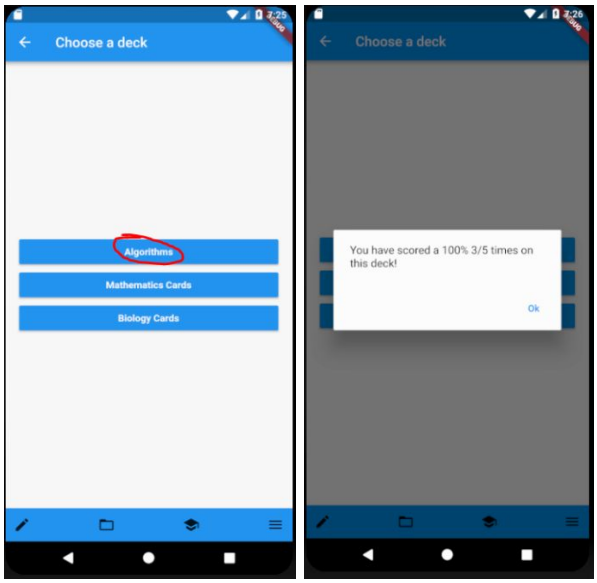
9. Recent Deck Access

- Users may access their recent decks by tapping the “Manage Flashcards” button. This will take them to a screen where they can access each individual deck.



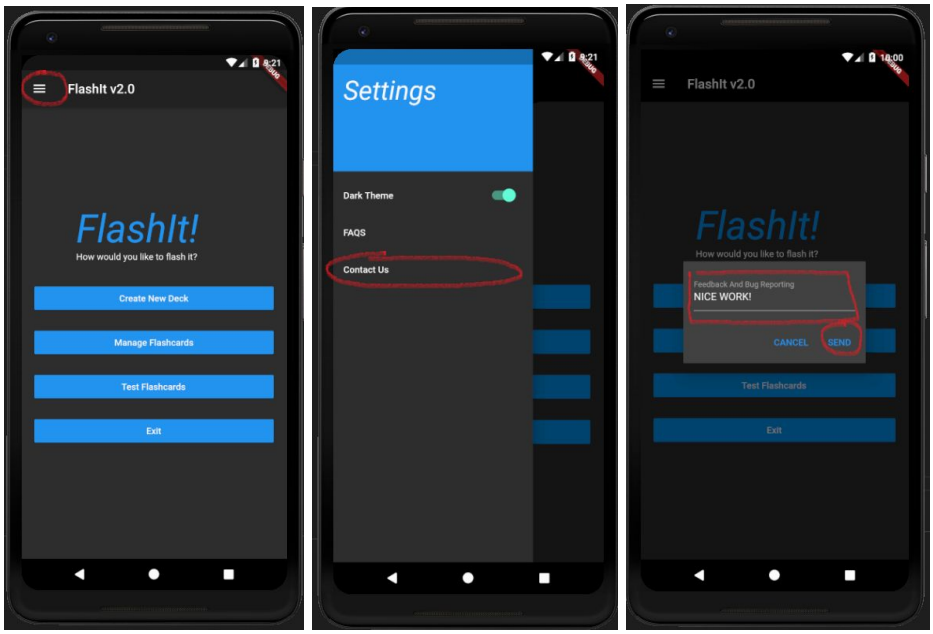
10. Deck Logistics

- Users may view information regarding their performance on certain decks. At the “View Sets” screen, the user may long press a set to view their history.



11. Bug Reports/Feedback

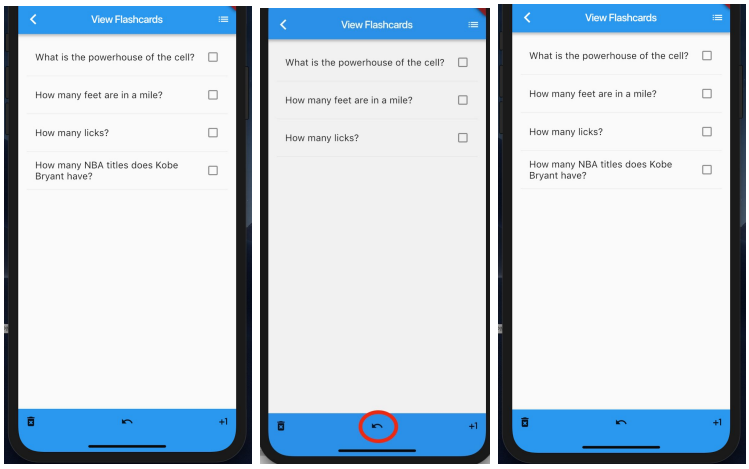
- Users may view information regarding their performance on certain decks. At the “View Sets” screen, the user may long press a set to view their history.



12. Undo Delete

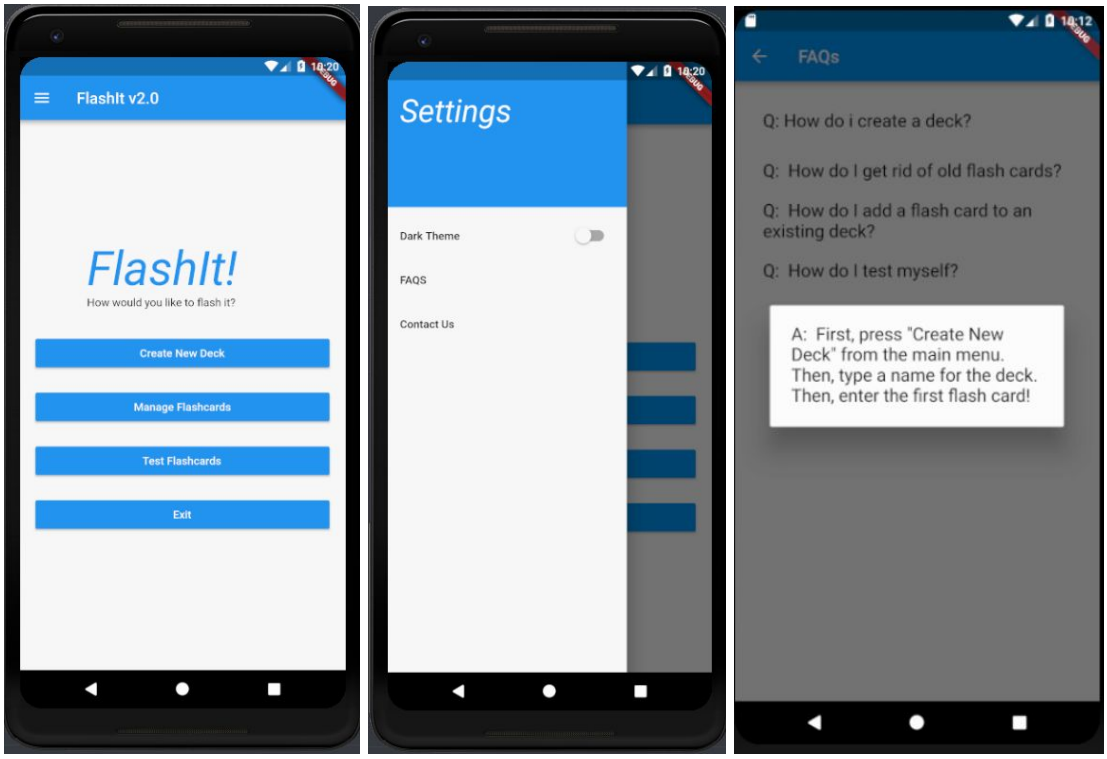
- Users may delete selected slides and restore them by clicking the undo after the deck has been deleted, be before exiting the deck management page.

-



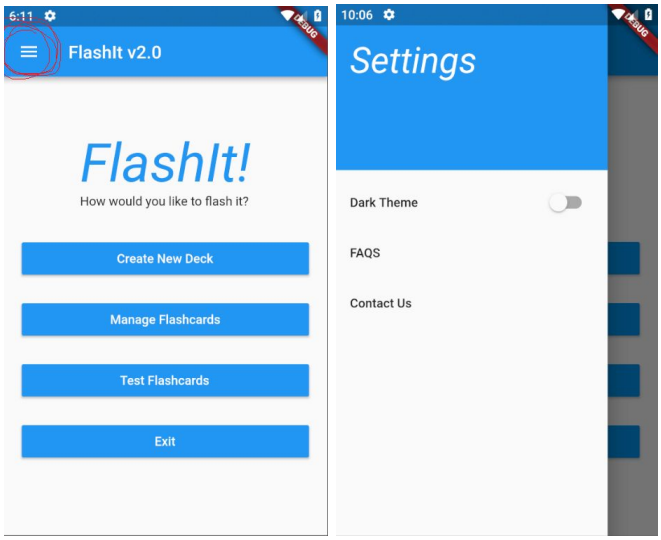
13. FAQs

- Users may view frequently asked questions for directions and guidance for the app's basic features.



14. Side Bar

- Users may view information regarding their performance on certain decks. At the "View Sets" screen, the user may long press a set to view their history.



15. Brightness Settings

- Users may change the brightness by accessing the sidebar, and sliding the toggle on the dark theme switch.