Locke (Virtual Lock Assistant)

Final Design document

Phase 1

John Herrera | CSE480HA (SU-18) | 8/01/2018

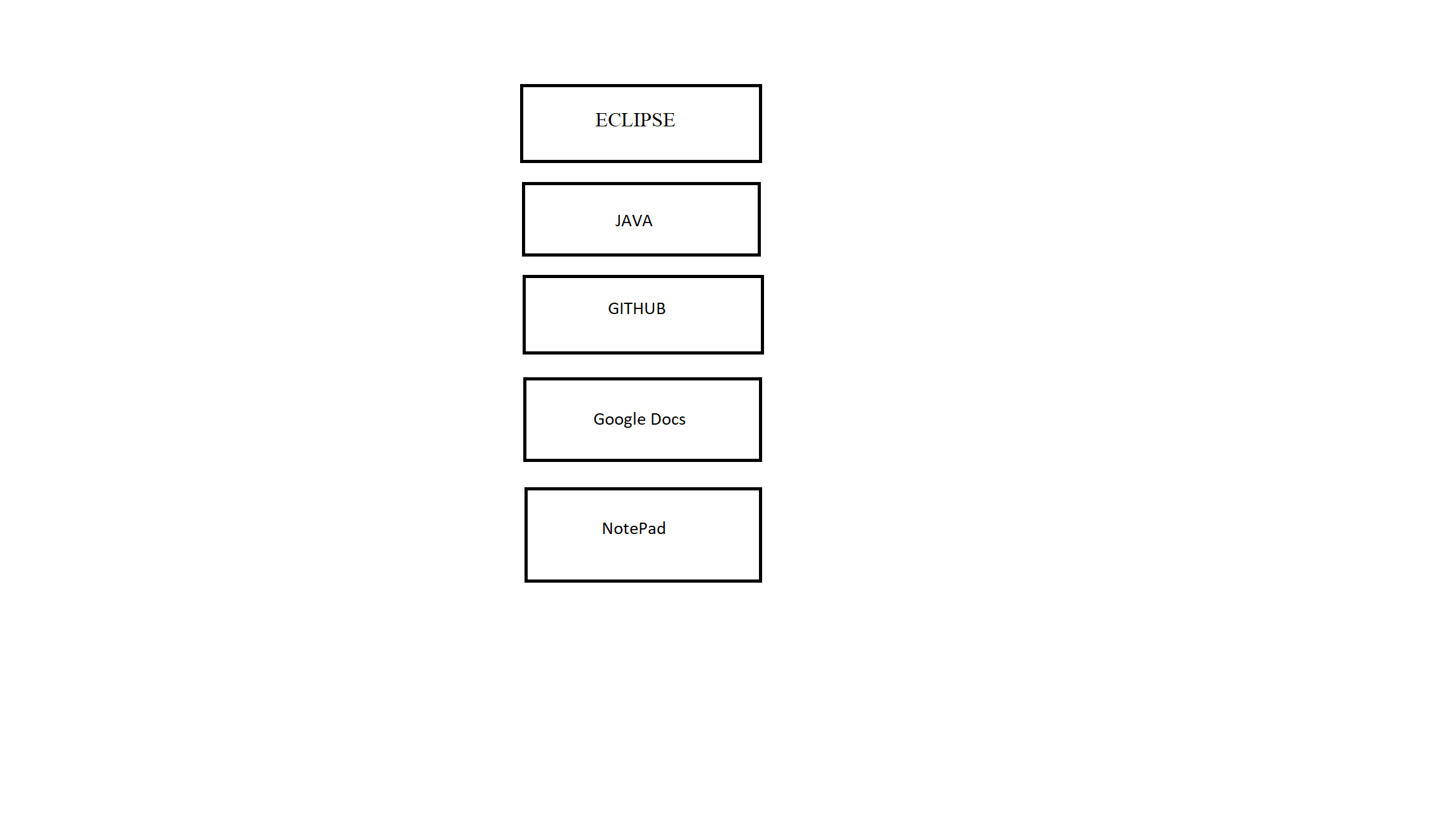
# REQUIREMENTS

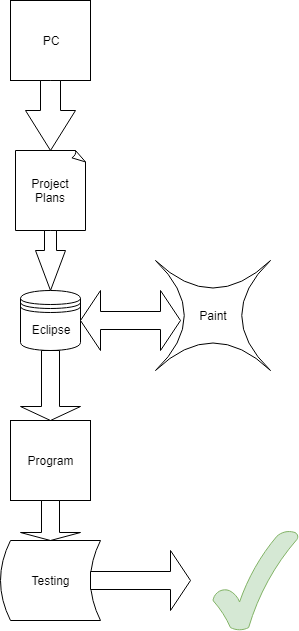
* **Project Description:** Locke is a virtual assistant which will generate and store passwords for the users based on their input. The application will have multiple pages ranging from the chat box to the passwords page.
* **Purpose:** This program is meant to generate a password for the user to use on a website then store it in a password page located in the application. Afterwards, the user can ask for the password for a certain site and Locke will tell you it. This is an easy way for people to generate, retrieve, and update their passwords while not being connected online so that all your passwords are safe.

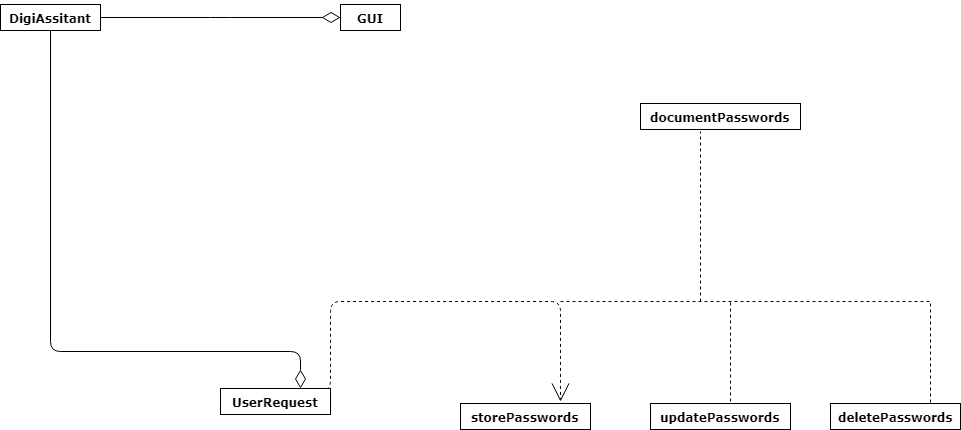
**Example:**

User: Generate password for BestBuy

Locke: A&CKADGF@15461 generated for BestBuy and stored

* **End Users of Application:** Students, workers, and people who forget their passwords in general.
* **Scope:** Will be a bot that responds to user input but return an error when non-supported request are inputted. Will only responds to greetings, generating requests, and already created passwords requests. Will have multiple pages that will have a chat box, passwords, and about page. The passwords will be encrypted so that others cannot see them if your device is stolen. Additionally, a login screen will be present for the users.
* **Functionality:** Will be created in Eclipse using java as the programming language.
* **Technology Stack:**
* **Architecture Diagram:**

****

* **Class Diagram/Component Level Diagram/Modules:**
* **Use Cases:**

Primarily used to generate passwords for the user to use.

Will also save passwords and provide to user when asked.

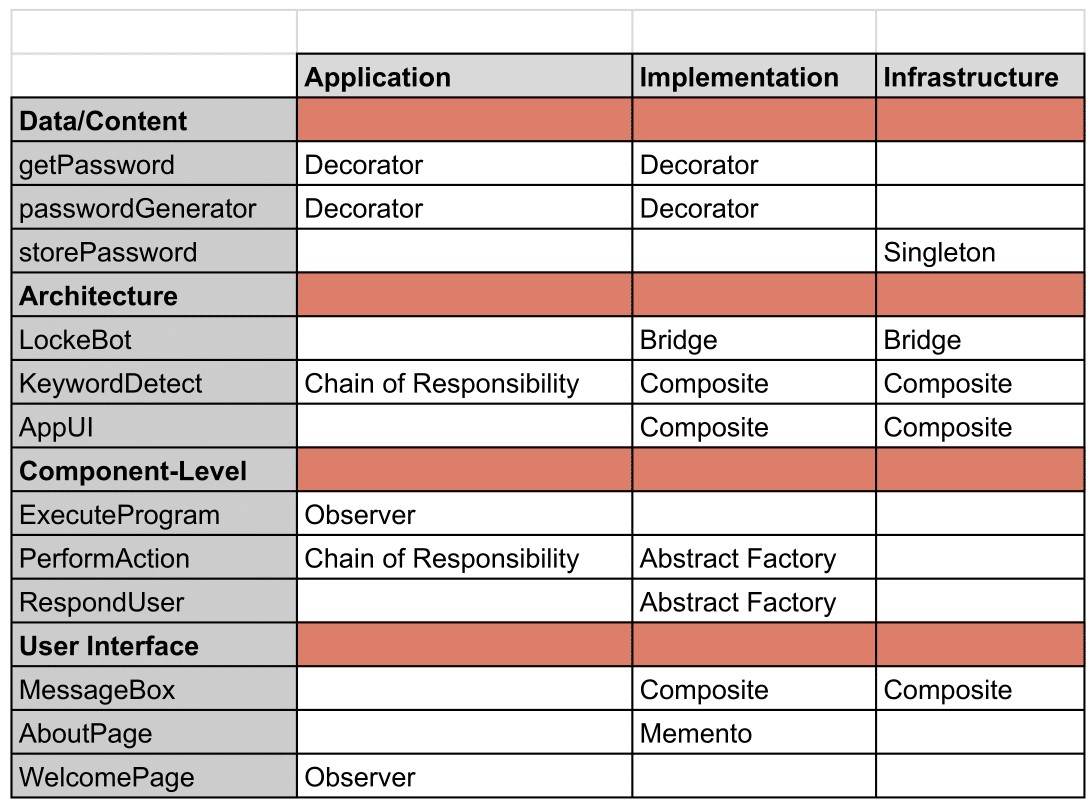
The passwords can be saved in a document by request.

The passwords can be destroyed by request.

The passwords can be updated by request.

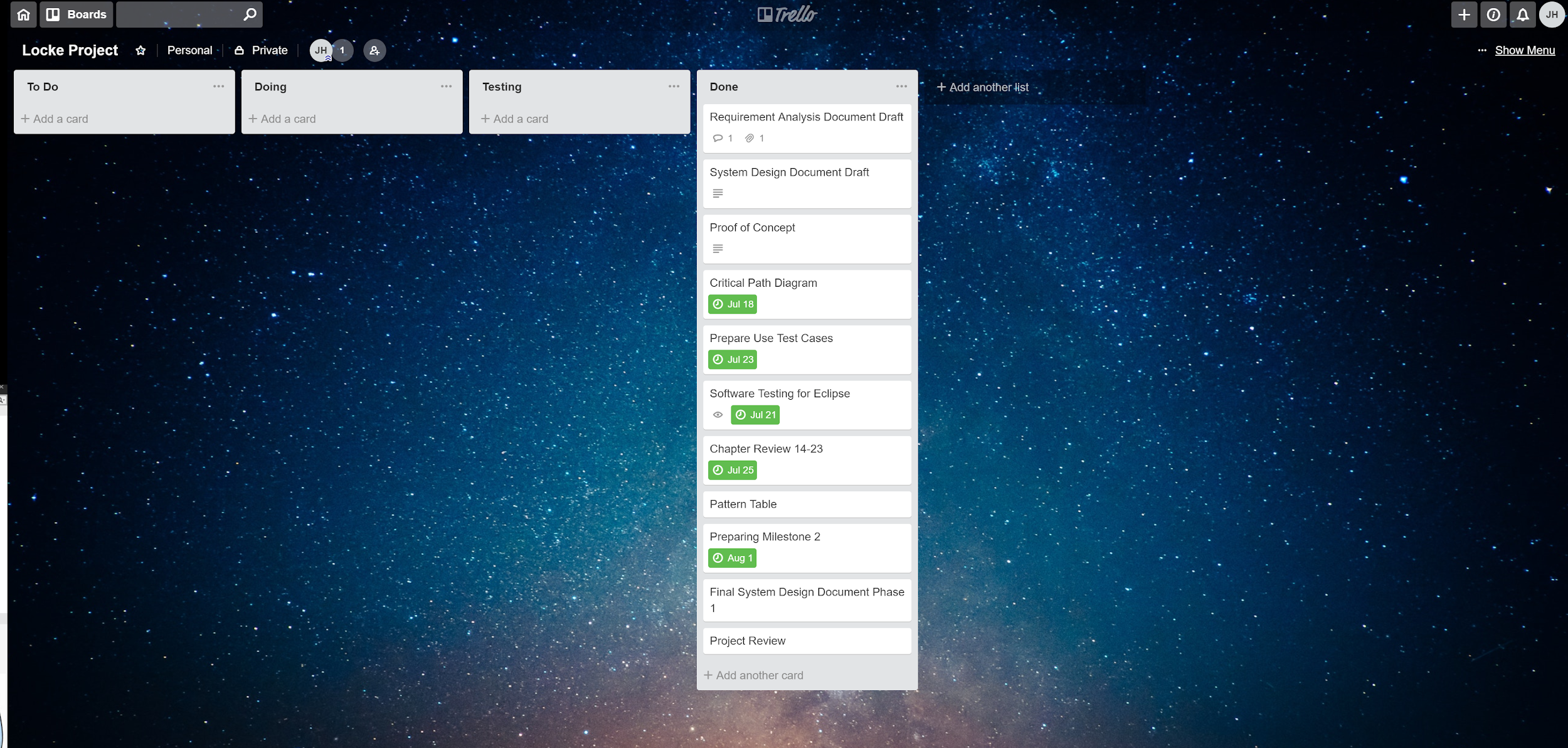
Can answer trivia related questions from users.

## Pattern organizing table



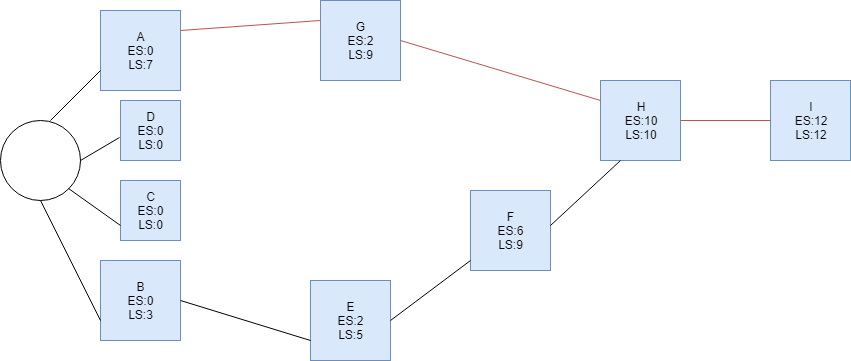
## Project planning board

* Requirement Analysis Document - Document describing our planned project- 4 Days
* System Design Document Draft - Document describing our project to be used in execution phase with more technical terms - 3 days
* Proof of Concept - Code example of our project - 5 days
* Critical Path - Document modulating our task that represent either our execution or planning phase of the project - 2 days
* Test Cases/Plans - Test cases and plans to bug test our application - 4 days
* Software Testing for Eclipse - Learning the Eclipse software for the execution phase - 2 days
* Chapter Review - 1 to 2 page assignment summary on book from professor - 1 day
* Pattern Table - Patterns that will be used for our statements when in execution phase - 3 days
* Milestone 1 - All assignments before midterm - 1 day
* Milestone 2 - All assignment planned to be done until final day in a text file - 1 day
* Final Design Document - Final draft of design document written in a professional manner - 5 days
* Project Review - Gathering all past assignments and getting a presentation ready for class - 5 days - Subtask: Powerpoint Presentation - 2 days



## critical path

**Activity Description Predecessor Duration**  
A UI programming (None) 2 Weeks  
B LockeAI programming (None) 2 Weeks  
C Test Cases (A, B) 1 Week  
D Documentation (None) 1 Week  
E Password programming (B) 4 Weeks  
F Encryption (E) 1 Week  
G Creating Executable (A,B,E,F) 1 Week   
H Bug testing (G) 2 Week  
I Project Finalization (H) 1 Week



## test cases

* **Black: passwordGenerator method:**   
  Input: Generate password for BestBuy  
  Return: A3sgaAs@12 generated for BestBuy and saved  
  Completes the imputed exercise  
  Detects keyword generate and then performs the action to create the password
* **White: updatePassword method:**Input: Update Amazon password to Lucky92@  
  Return: Password Updated  
  Completes imputed exercise  
  Detects keyword update to perform action on Amazon that is stored in a file and therefore changing password
* **Grey: lockeTriviaResponse method:**Input: Tell me your name  
  Return: My name is Locke!  
  Completed the imputed exercise  
  User requests trivial information from bot and responds with pre-saved responses created beforehand.
* **storePassword method.**Input: Store CleayQ123# for PSN  
  Return: CleayQ123# stored for PSN  
  Completed the imputed exercise  
  Detects keyword store and performs action on PSN which was added into file and then attaching user inputted password
* **deletePassword method.**Input: Delete Gamestop password  
  Return: Gamestop and attached password deleted  
  Completed the imputed exercise  
  Detects keyword delete and performs action on Gamestop which is stored on file and then removes it.
* **documentPassword method.**Input: Document passwords  
  Return: Passwords document into a text file   
  Completed the imputed exercise  
  Detects keyword document and performs action by documenting all passwords stored in application into a text file after being decrypted.