CheckersVision

Group 5:

-Charles Dudley
-Ranai Srivastav
-Siddhant Jaju
-Sibhat Yosef
-Yaaseen Basha

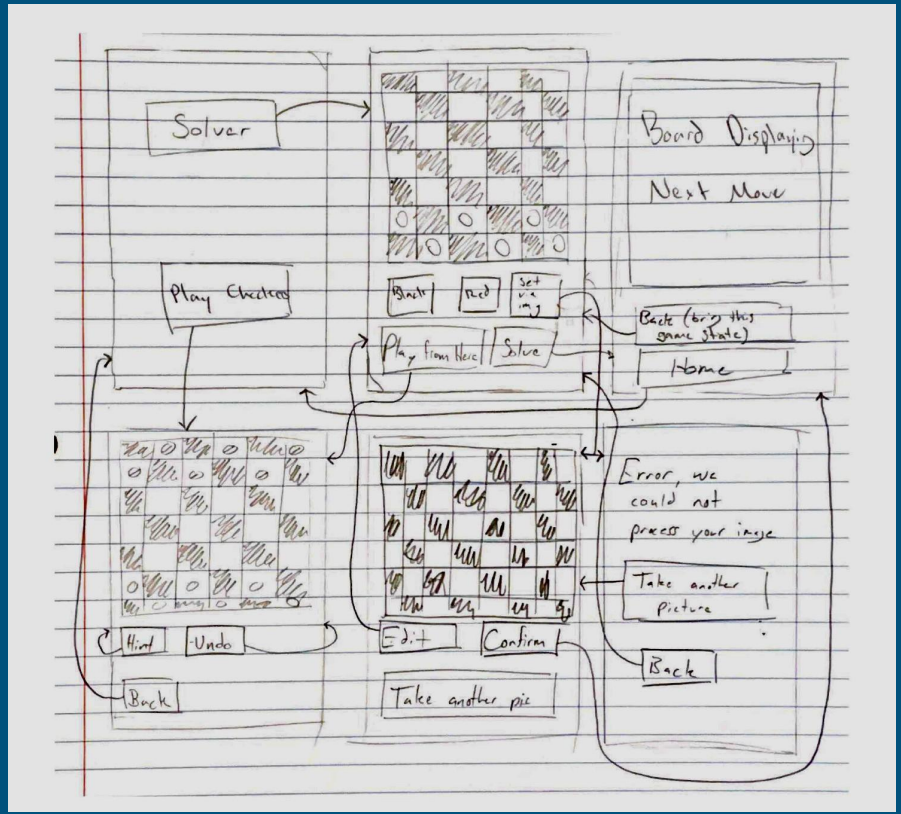
Purpose

- Resource for learning **Checkers**
  - Leverage checkers solver to enhance learning
    - Play against bot
    - **Solve** for next move
    - And more!
  - **Allow users to learn while playing over the board**
- Platform for our team to learn various technologies of interest
  - **Computer Vision**
  - **Frontend** development
  - **Cloud** computing

High Level Walkthrough

- Cross-platform mobile app using **Flutter**
- **GUI** to allow the user to play and learn strategy of Checkers
- Use **computer vision** to recognize a game state
  - Allow user to solve optimal next move in over the board game
- Serverless backend hosted on **AWS**

High Level Walkthrough cont.



Frontend : Flutter/Dart

What is **Flutter**?

- Open-source SDK used for cross platform mobile development

Why Flutter over other software?

- Something new and interesting to learn
- Can build native mobile apps quickly for both **iOS** and **Android** with a single codebase using the **Dart** language
- Hot reload feature
    - Allows the developer to make any changes to the app while its running.

Front-end Development Tools

Visual Studio Code

- Source code editor
- Great extensions to aid in flutter development
- Strong interface for debugging
- Good support for Git

Android Studio

- IDE for Android development
- Useful for deploying app on Android Phones

Xcode (Mac Only)

- IDE For iOS development
- Needed for deploying app on iOS

Mobile Phones

- iPhone/Android

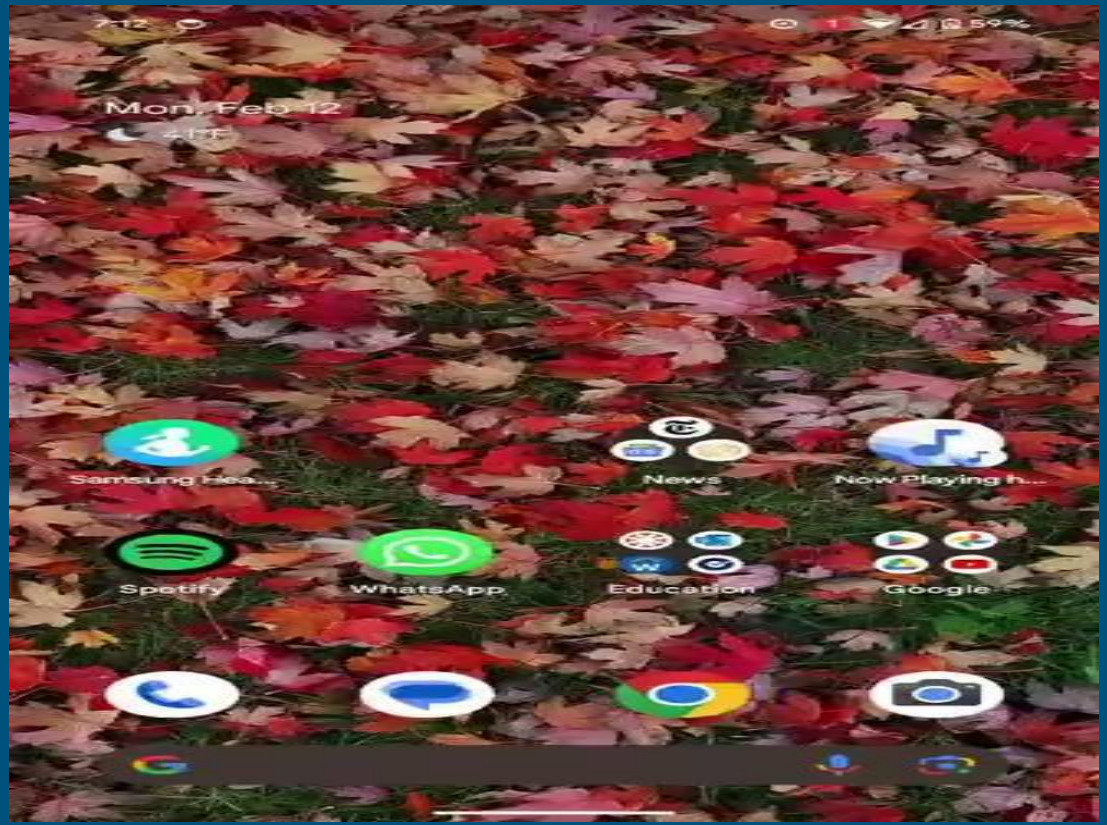
Anticipated Challenges

General Flutter challenges

- Learning curve: No previous experience with Flutter/Dart
- Optimize **user experience**
  - Checkers simulator should be **easy** to play
  - **Navigation** should be **straightforward**
- **State management**
  - Keeping track of data throughout the app
  - Looking into REDUX
- Platform specific issues
  - Maintaining a consistent user experience iOS and Android

Photo Library App Demo

Photo App Demo



Early Game Setup/Play Demo

Checkers Engine

What is a **Checkers Engine**?

- Uses **search algorithms** and different computational strategies to evaluate game state and compute an optimal move for the player.

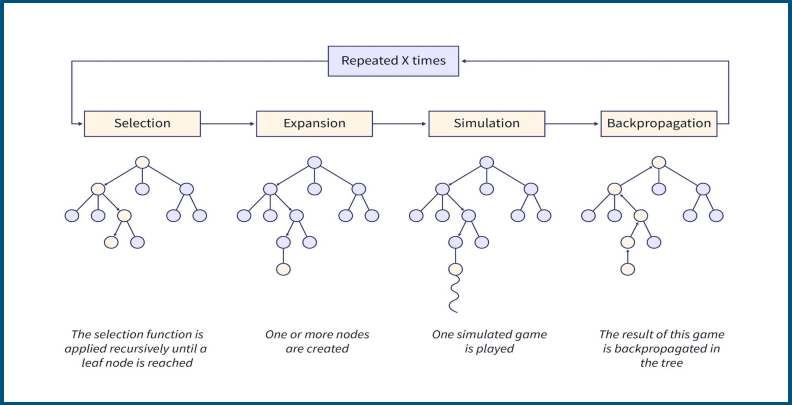
Why do we need it?

- It plays a role in user understanding of the strategy and problem solving in Checkers.

Implementation

- **Board Representation**: 2-d array where each cell represents a square on the board, indicates whether empty, black or red piece or is a king.
- **Game Logic**: Rules for moving pieces, capturing opponent pieces, promoting pieces to kings and determining when the game is over.
- **Valid Move Generation** : Generate all possible valid moves for the player given the current state of the board.
- **Move Execution:** Implement functions to apply a move to the game board and update the game state accordingly.
- **Game State Management and Outcome** : Keeps track of the current player, number of pieces remaining and game ending conditions.

Handling Checkers Data



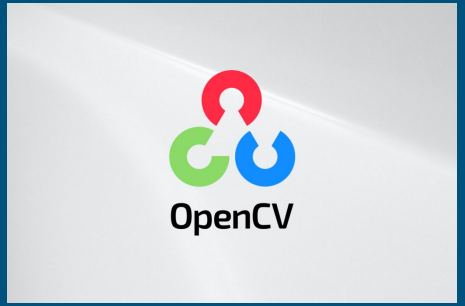
The main objective here is to implement the ‘Monte-Carlo Tree Search’ algorithm for handling the checkers engine.

- MCTS Selection - We use a strategy to choose which nodes to explore in the tree. In checkers, factors like number of visits and win rates when selecting nodes will be considered.
- MCTS Expansion - The tree will be expanded by adding child nodes for unexplored moves from the selected node.
- MCTS Simulation - Simulate random moves from the newly added child nodes to estimate their potential.
- MCTS Backpropagation - Update the statistics of all nodes along the path from the root to the leaf based on the results of the simulated moves. This will include updating visit counts and win counts.

Functionalities

- List all possible moves
- Check if a move is valid
- Make a move
- Run a Monte Carlo Tree Search simulation for each move
- Check if the game is over

OpenCV



What is **OpenCV**?

- **Computer Vision** and **Machine Learning** software library related to image and video analysis

Why use OpenCV?

- Cross platform functionality, Open Source, extensive Documentation, etc.
- Implement **board and piece detection** to serialize game state for **Checkers Engine**

CV steps

Steps of Implementation (first on screenshots):

- Detect board
  - findChessboardCorners() method exists, but some additional parameter tuning will need to be done
  - Corner detection and line detection using methods such as goodFeaturesToTrack()
- Piece detection
  - Need to decipher what pieces are where
- Store information
  - Need to store this information into 2-D array to be passed into Engine

Checkerboard detection (built in method)

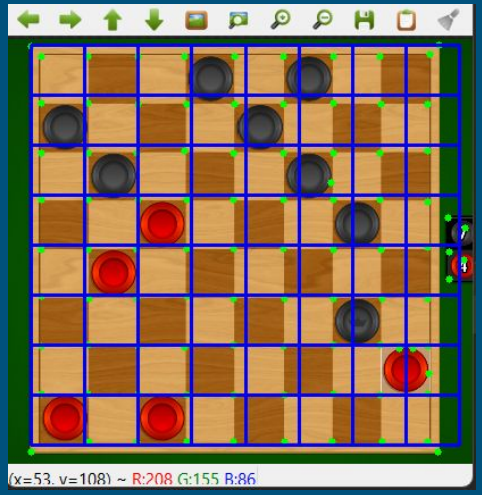
Using built in findChessboardCorners - of the 4 screenshots, worked on 1

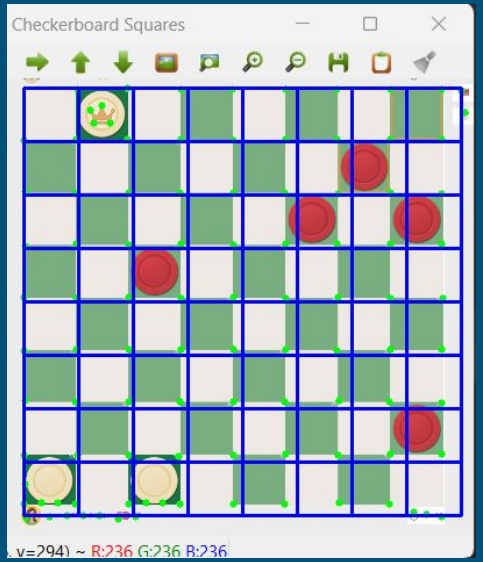
Not as robust, lot of fine tuning still needs to be done as it only detects the inner 7x7 board and additional computation is required to detect the full edges of the board, thus making it error prone.

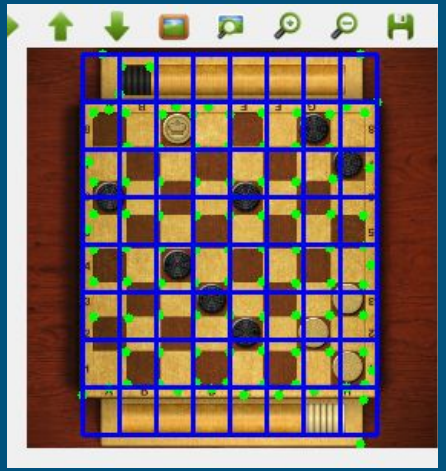
Basic Corner and Line Detection Steps

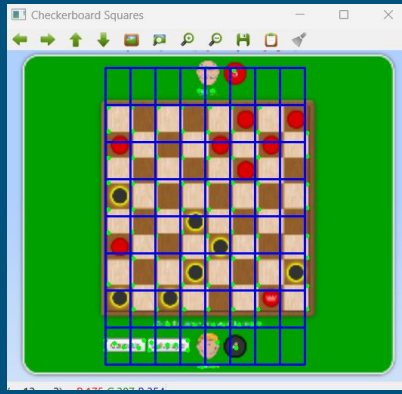
1. Find corners of the board
2. Sort corners in a consistent order
3. Find boundaries by finding minimum and maximum x and y coordinates
4. Divide into an 8x8 grid
5. Identify square corners

Checkerboard detection (corner & line detection)





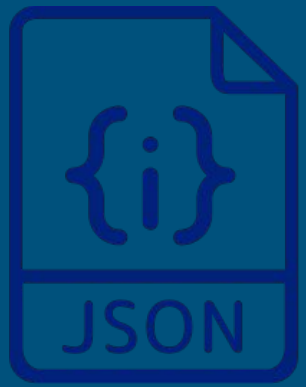




Anticipated Challenges with CV

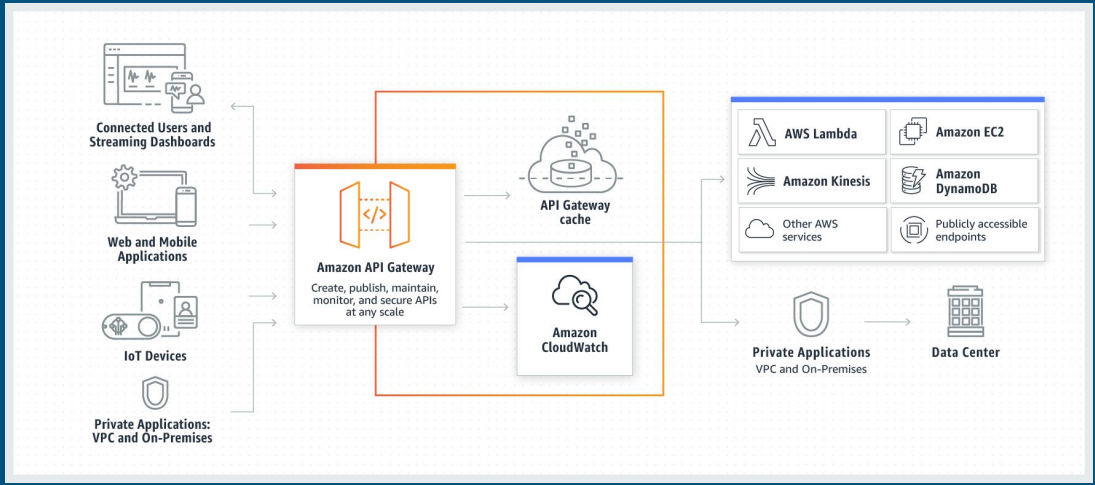
- Variability - Robust detection for screenshots and real life boards
- Identifying king pieces
- Processing time - CV can get slow
- Integration with other parts

Backend: Amazon Web Services



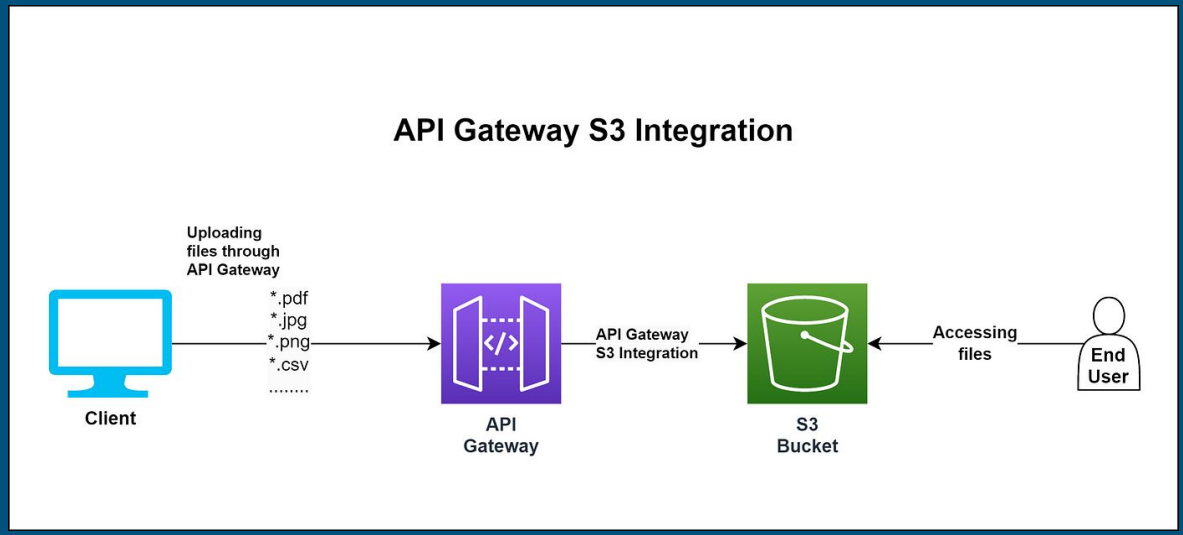
API Gateway

- Develop **secure APIs** at scale
- Allow the user to upload image
- Provide a way to move information from frontend to backend



Simple Storage Service

- Used as a **datastore** for cloud native applications (CDN)
- **Efficient** function calls



Lambda

- Used for **serverless** functions,
- **Efficient** function calls
- Hosts the **Checkers Engine** logic

AWS

API Gateway to get S3 upload token

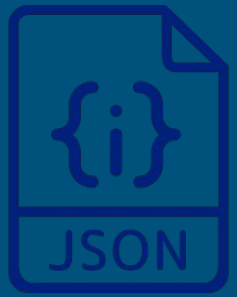
Store Image in **S3** bucket

Lambda function for serverless method calls

API Gateway returns best move as **JSON** to app

Amazon API Gateway

AWS Lambda



Anticipated Challenges

- AWS learning curve
- Not enough knowledge of AWS products
- Integration between modules

Timeline

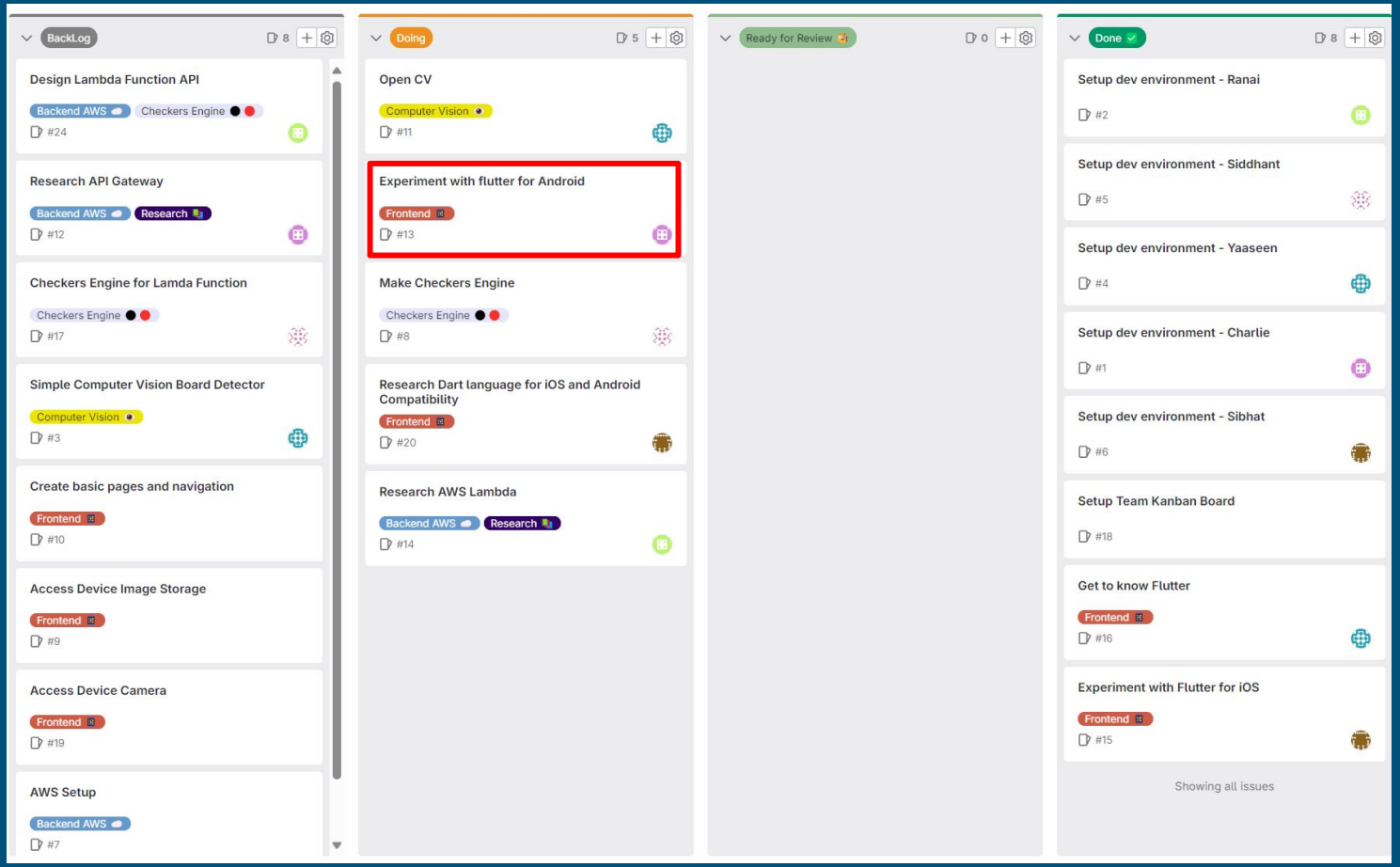
Demo-1:

- Setup development environment.
- Learn Flutter, Experiment with Flutter for iOS and Android.
- Get familiar with Open CV.
- Plan the Checkers Engine.
- Research about AWS Lambda.
- Access device camera and gallery.

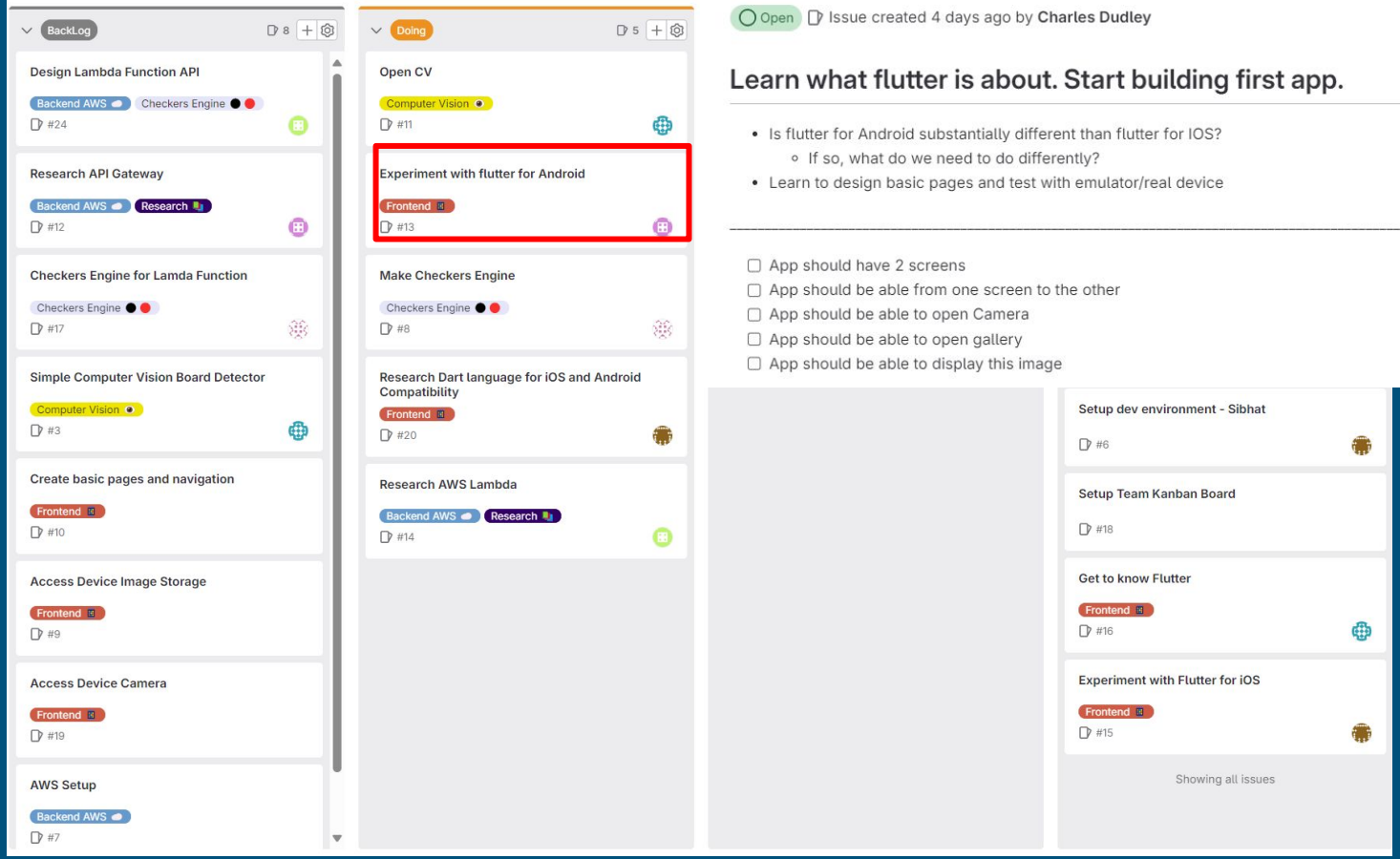
Demo-2:

- Fully implement the Checkers Engine.
- Implement some functionality using AWS Lambda.
- Fully implement the Computer Vision Board Detector.
- Partially implement the GUI interface.

Timeline


BackLog
8
Design Lambda Function API
Backend AWS
Checkers Engine
#24
Research API Gateway
Backend AWS
Research
#12
Checkers Engine for Lamda Function
Checkers Engine
#17
Simple Computer Vision Board Detector
Computer Vision
#3
Create basic pages and navigation
Frontend
#10
Access Device Image Storage
Frontend
#9
Access Device Camera
Frontend
#19
AWS Setup
Backend AWS
#7
Doing
5
Open CV
Computer Vision
#11
Experiment with flutter for Android
Frontend
#13
Make Checkers Engine
Checkers Engine
#8
Research Dart language for iOS and Android Compatibility
Frontend
#20
Research AWS Lambda
Backend AWS
Research
#14
Ready for Review
0
Done
8
Setup dev environment - Ranai
#2
Setup dev environment - Siddhant
#5
Setup dev environment - Yaaseen
#4
Setup dev environment - Charlie
#1
Setup dev environment - Sibhat
#6
Setup Team Kanban Board
#18
Get to know Flutter
Frontend
#16
Experiment with Flutter for iOS
Frontend
#15
Showing all issues

Timeline


BackLog
8
Design Lambda Function API
Backend AWS
Checkers Engine
#24
Research API Gateway
Backend AWS
Research
#12
Checkers Engine for Lamda Function
Checkers Engine
#17
Simple Computer Vision Board Detector
Computer Vision
#3
Create basic pages and navigation
Frontend
#10
Access Device Image Storage
Frontend
#9
Access Device Camera
Frontend
#19
AWS Setup
Backend AWS
#7
Doing
5
Open CV
Computer Vision
#11
Experiment with flutter for Android
Frontend
#13
Make Checkers Engine
Checkers Engine
#8
Research Dart language for iOS and Android Compatibility
Frontend
#20
Research AWS Lambda
Backend AWS
Research
#14
Experiment with flutter for Android
Edit
Open
Issue created 4 days ago by Charles Dudley
Learn what flutter is about. Start building first app.
Is flutter for Android substantially different than flutter for IOS?
If so, what do we need to do differently?
Learn to design basic pages and test with emulator/real device
App should have 2 screens
App should be able from one screen to the other
App should be able to open Camera
App should be able to open gallery
App should be able to display this image
Setup dev environment - Sibhat
#6
Setup Team Kanban Board
#18
Get to know Flutter
Frontend
#16
Experiment with Flutter for iOS
Frontend
#15
Showing all issues

Questions?

- CheckersVision
- Flutter/Dart
- Android/iOS
- OpenCV
- Checkers Engine
- Serverless Application
- AWS
  - API Gateway
  - S3
  - Lambda