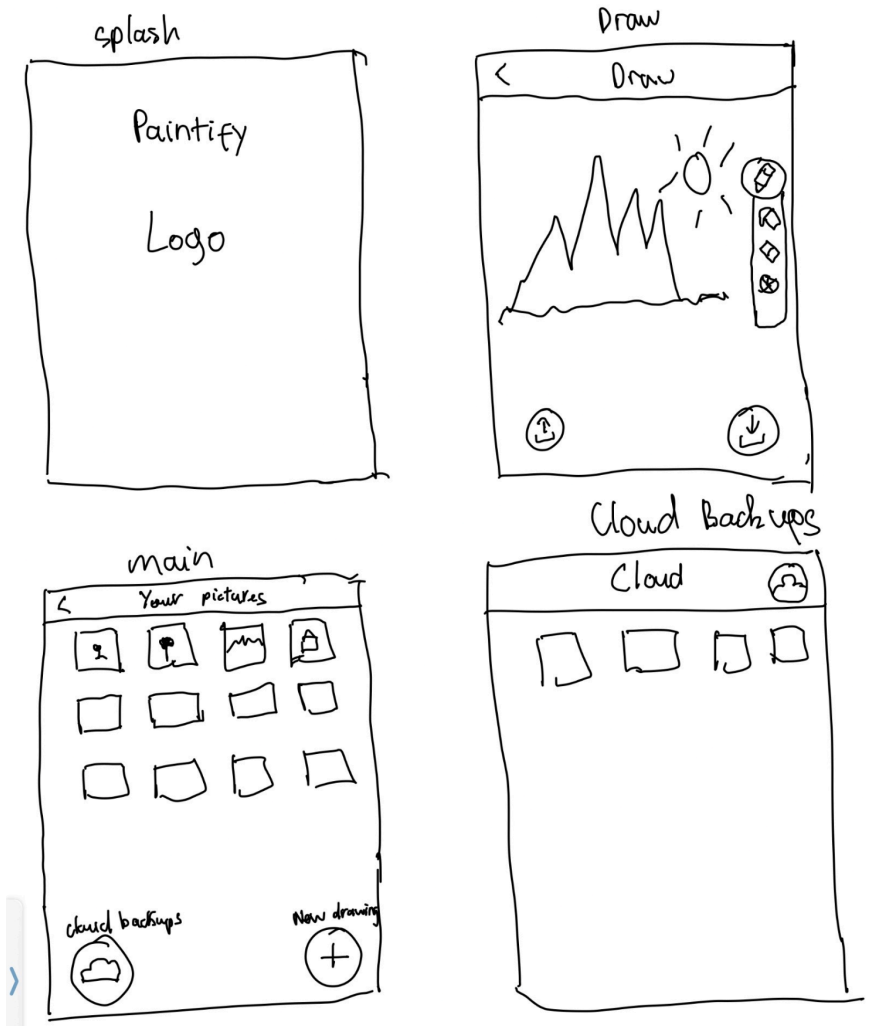
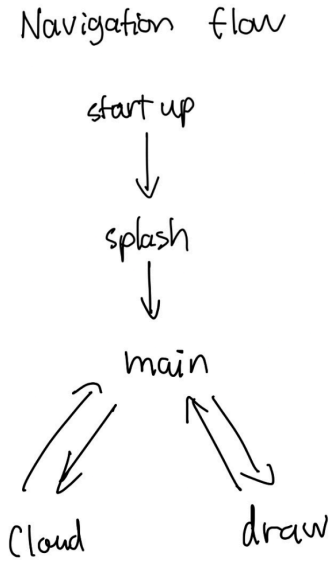
Paintify - Phase 1 Implementation Plan - Dustin, Nolan, IAN



* delete by holding



Layouts (screens) to create (Phase 1)

1. Splash screen should be there by default (we'll change the Android logo to our custom logo)
2. `DrawActivity -` like the drawing of the "Draw" sceen above, it contains a full-screen canvas with toolbar on the right edge of the screen (Pen, Eraser, Color Palette, etc). And the save button at bottom right corner. The left button was intended for uploading a saved picture and make it editable on the canvas.

`NOTE: MainActivity –` contains two buttons to navigate to DrawActivity, CloudActivity, and will contain the locally saved previews of the user's drawn pictures (deferred to Phase 2 since it contains the saving method and `CloudActivity`)

Classes to implement (Phase 1)

UI

- `SplashActivity`

- `DrawActivity`

- `DrawScreen` (Composable): contains right-edge tool rail, bottom save button, and the canvas area.
- `DrawCanvas` (Composable): the canvas area to render and capture strokes.
- `ToolRail`: buttons for choosing `ToolType` (enum)
- `BrushSizePicker` – Composable slider + live preview when changing the brush size
- `ColorPicker` – a small Composable component of preset swatches (e.g., black, white, red, green, blue, yellow, purple) that will appear when the “choose color” tool is touched.

State (MVVM)

- `DrawViewModel (VM in MVVM)`:
    - `currentTool` (enum: PEN, LINE, RECT, CIRCLE, ERASER)
    - `penColor` (int)
    - `penSizePx` (float)
    - `strokes` (immutable list of `Stroke` / `ShapeCommand`)
    - `redoStack`
    - intent methods: `beginStroke()`, `extendStroke()`, `endStroke()`, `undo()`, `redo()`, `setColor()`, `setSize()`, `setTool()`

Models

- `ToolType` (enum)
- `Stroke` (data: list of points, paint params)
- `Shape` (enum: `LineShape`, `RectShape`, `CircleShape`)
- `PaintParams` (color, width, style)

Unit tests to write (Phase 1)

test/.../DrawViewModelTest

- Tool selection updates correctly (including `ERASER`).
- Color/size setters update state (`penColor`, `penSizePx`).
- Stroke lifecycle: `beginStroke → extendStroke → endStroke` stores a stroke with expected points and `PaintParams`.
- Shapes: for tools `LINE / RECT / CIRCLE`, `begin → end` creates the correct shape with derived bounds.
- Undo/Redo behavior:
    - Undo removes the last committed stroke; redo restores it.
    - New stroke clears `redoStack`.
    - No crash on empty undo/redo.
- Eraser semantics: strokes created while `currentTool = ERASER` are flagged (e.g., `isEraser = true`) and carry expected `PaintParams`.

test/.../PaintParamsTest

- Width and color propagate to newly created strokes/shapes.
- Eraser paint flag/params are set as expected (e.g., `isEraser = true`).

androidTest/.../SplashScreenTest

- Splash logo is displayed.

androidTest/.../SplashNavigationTest

- App launches, shows splash, navigates to DrawActivity.

androidTest/.../DrawUiSmokeTest (Compose)

- Tool rail buttons render; tapping Color and Brush Size pickers opens and selects values (no crash).
- Choosing a tool, performing a drag on DrawCanvas increases the ViewModel stroke count (read via a test hook or exposed state).
- Switching to Eraser and drawing produces an eraser-typed stroke in state (verify `isEraser` or equivalent).

Implementation order (Phase 1)

Day 0–1: Setup

- Repo + branch protections, all members & TAs added.
- Project skeleton (single app module), base theme.
- SplashActivity → DrawActivity navigation (DrawActivity uses `setContent { DrawScreen() }`).

Day 1–2: MVVM skeleton

1. Create DrawViewModel with state fields and basic setters:
    - `currentTool`, `penColor`, `penSizePx`, `strokes`, `redoStack`.
    - Intents: `setTool`, `setColor`, `setSize`, `undo`, `redo`.
2. Create DrawScreen (Composable) scaffold:

    - Right-edge ToolRail (buttons only, no behavior yet).
    - Bottom Save button (stubbed).
    - Center DrawCanvas placeholder.
3. Wire DrawScreen to DrawViewModel.

Day 2–4: Drawing core

4. Implement DrawCanvas (Composable):
    - `Canvas {}` + `pointerInput { detectDragGestures(...) }`.
    - Maintain in-progress stroke locally (preview path while dragging).
    - On drag start → `beginStroke(x,y)`; on move → `extendStroke(...)` (optional throttle); on end → `endStroke()`.
    - Render committed strokes from DrawViewModel.
5. Render shapes (`LINE/RECT/CIRCLE`) with live preview while dragging; commit on `endStroke`.
6. Wire Undo/Redo buttons to ViewModel.

Day 4–5: Customization

7. Implement ColorPicker (Composable) and BrushSizePicker (Composable):
    - Hook selections to `setColor()` and `setSize()`.
8. Tool selection buttons set `currentTool` (including `ERASER`).

Day 5–6: Tests & polish

9. Write/finish JVM unit tests:
    - `DrawViewModelTest`, `PaintParamsTest` (items listed above).

10. Write instrumented tests:

- `SplashNavigationTest`, Compose `DrawUiSmokeTest`.

Task assignment (Phase 1)

Nolan

- Implement **DrawCanvas**'s gesture handling with `pointerInput`, local in-progress path, and commit to **DrawViewModel (VM)** on `endStroke()`.
- **Shape drawing**: live preview for `LINE/RECT/CIRCLE` while dragging; commit to VM on release.
- **Eraser** rendering
- **Unit tests**: `DrawViewModelTest` — stroke lifecycle (begin/extend/end), shapes (bounds correctness).

Dustin

- Implement/integrate **DrawViewModel** (state, intents, redo invariants)
- Build **ColorPicker** and **BrushSizePicker** Composables; hook selections to VM's `setColor()` / `setSize()`.
- Bind **DrawScreen** to VM, ensure tool/color/size reflect state.
- **Unit tests**: tool selection, color/size setters, undo/redo edge cases (empty stacks, **new stroke clears `redoStack`**), `PaintParams`.

Ian

- **SplashActivity** (1–1.5s delay or immediate route) and app theming.

- **Undo/Redo** wiring from **ToolRail** to VM intents.
- **DrawActivity** scaffold hosting `setContent { DrawScreen() }`; layout of **ToolRail**, bottom **Save** button, and canvas region.
- **Tests**: splash navigation, Compose UI smoke (open pickers, draw a stroke).