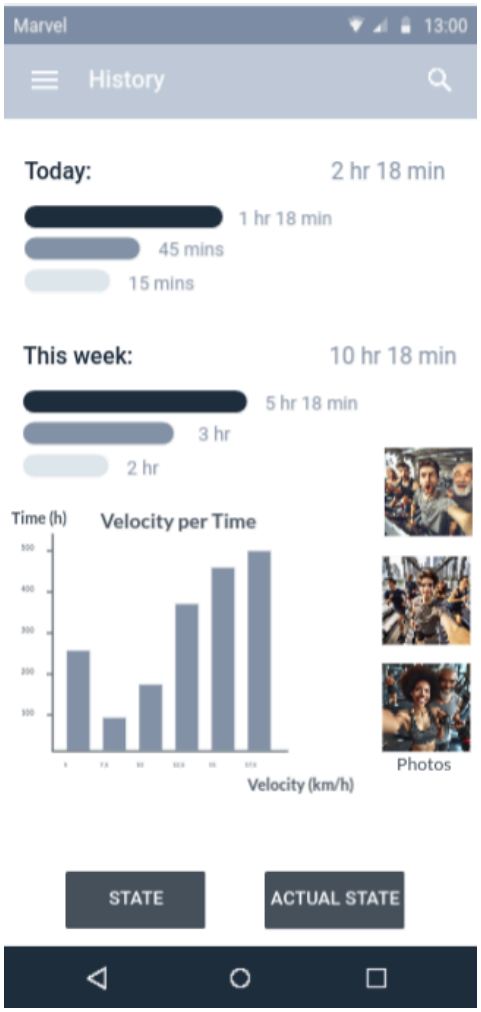
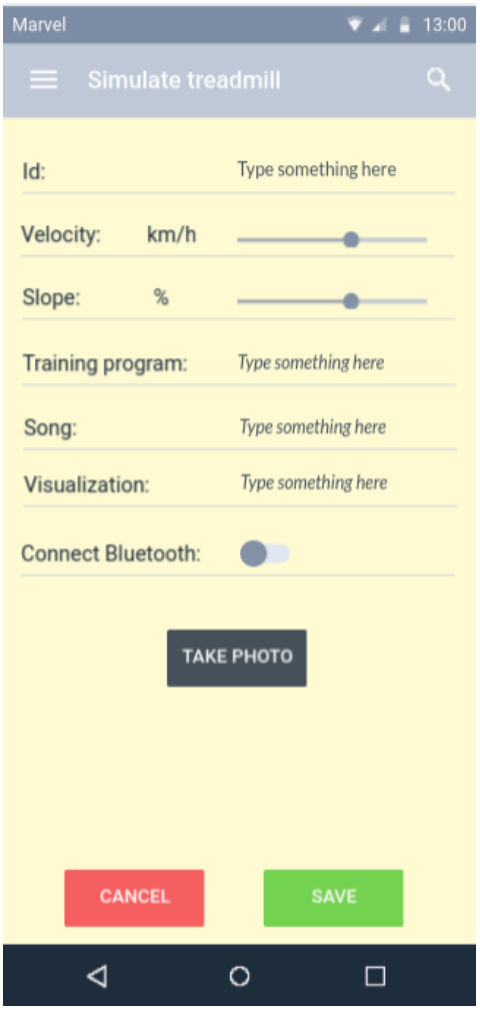
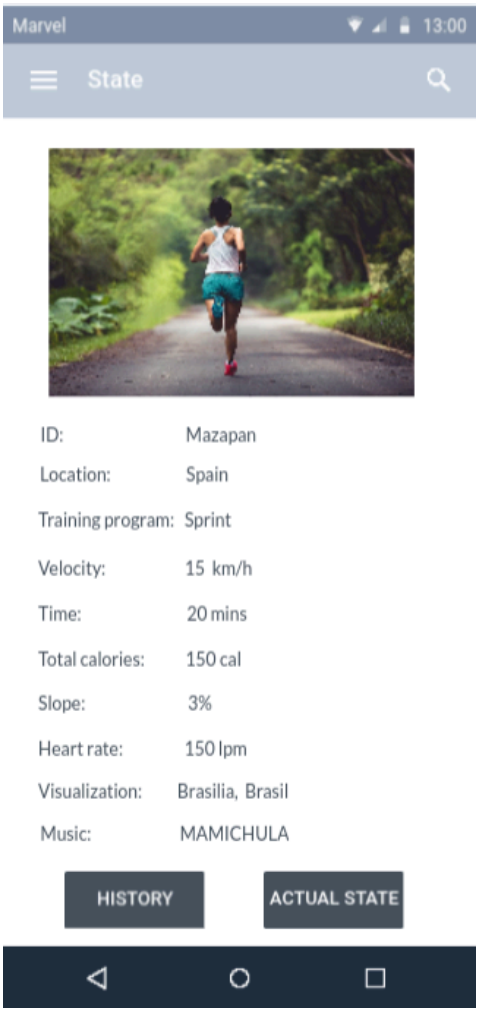
Basic design: 3 user interface screens, made with marvelapp or powerpoint, includes web link to marvelapp page.

LINK: https://marvelapp.com/prototype/10a5045b







Input touch functionality implementation

1. **Design UI**: Use Android Studio to create buttons and sliders for speed, incline, music control, and training modes.
2. **Touch Event Handling**: Add touch event listeners to detect user interactions like tapping, swiping, or dragging.
3. **Control Features**: Enable users to adjust speed, incline, and music playback by tapping or swiping on the screen.
4. **Select Modes**: Allow users to choose training modes by tapping icons or buttons representing different programs.
5. **Feedback and Testing**: Provide visual feedback for user actions and thoroughly test the touchscreen interface for responsiveness and usability.
6. **Accessibility**: Ensure accessibility by making touch targets large and considering alternative input methods.

Implementing Camera Functionality for Capturing Post-Run Moments

1. **Trigger Mechanism**: Implement a button or gesture-triggered mechanism to activate the camera after the user completes their run.
2. **Capture Image**: Enable the software to capture a photo using the device's camera at the moment the user initiates the action.
3. **Save Image**: Store the captured image locally or in cloud storage for the user to access later.

By implementing these steps, users can capture and preserve their post-run moments seamlessly within the treadmill software.

In this application, I will employ an SQL database to record and manage the historical data of treadmill sessions. This database will serve as a repository for crucial metrics such as speed, incline, training program, duration, calories burnt, heart rate, and more. Furthermore, alongside storing numerical data, I will integrate functionality to save images of each user upon concluding a session. By structuring the data within an SQL database, we ensure efficient organization, retrieval, and analysis of treadmill usage over time.

Model Classes:

MachineStatusModel:

- Attributes: id, location, training program, velocity, time, total calories, slope, heart rate, visualization, music
- Methods: Getters and setters for each attribute

TrainingProgram:

- Attributes: id, name
- Methods: Getters and setters for each attribute

View Interfaces:

MachineStatusView:

- Attributes: id, location, music, program, heart rate, velocity, slope, time, visualization
- Methods:
    - displayMachineStatus(MachineStatusModel machineStatus)
    - simulateTreadmill()
    - watchHistory()

MachineSimulationView:

- Attributes: velocity, slope, training program, song, visualization, connectionBluetooth
- Methods:
    - displayMachineSettings(MachineStatusModel machineStatus)
    - saveSettings()
    - cancelSettings()
    - takePhoto()

TrainingUsageGraphView:

- Methods:
    - getters and setters
    - obtainDataFromDB()
    - displayTrainingUsageGraph()
    - simulate()
    - watchState()

Data Access Object (DAO) Classes:

MachineStatusDAO:

- Methods:
    - fetchMachineStatus(): MachineStatusModel
    - updateMachineStatus(MachineStatusModel updatedStatus)

TrainingProgramDAO:

- Methods:
    - fetchAllTrainingPrograms(): List<TrainingProgram>
    - fetchTrainingProgramById(int id): TrainingProgram
    - updateTrainingProgram(TrainingProgram program)