**User Manual**

**Introduction**

The Yapp application allows users to create unique media experiences by combining audio, images, and video. Users can:

1. Capture or select an image.
2. Record or upload audio (up to 2 minutes).
3. Combine the image and audio to create a video (Yapp).
4. Play or share created Yapps.

**Features**

* **Create Yapp**: Combine audio and images into videos.
* **Manage Yapps**: Rename or delete Yapps.
* **Share**: Share Yapps through various platforms.

**Instructions**

**Creating a Yapp**

1. Open the application.
2. Tap "Create a Yapp" or press the "+" floating button.
3. Select or capture an image.
4. Record audio (maximum 2 minutes) or upload an audio file.
5. Tap "Create Yapp" to generate your video.
6. Use the "Play Yapp" button to view the video.

**Managing Yapps**

1. Long-press on a Yapp to enter selection mode.
2. Rename a selected Yapp using the edit icon.
3. Delete one or more Yapps using the trash icon.

**Sharing Yapps**

1. Tap the share icon next to a Yapp to share it through available platforms.

**Technical Guide**

**Architecture**

The Yapp application uses a **provider-based architecture** for state management. It integrates:

* **Provider Package**: Manages app state (YappProvider).
* **FFmpegKit**: Handles audio and video processing.
* **VideoPlayer**: Plays created Yapps.
* **Flutter Widgets**: Builds the user interface.

**Key Components**

1. **Models (yapp.dart)**: Represents Yapps with attributes such as ID, name, image path, audio path, video path, and creation date.
2. **Provider (yapp\_provider.dart)**:
   * Adds, removes, and renames Yapps.
   * Generates default Yapp names.
3. **Screens**:
   * **YappListScreen**: Displays a list of Yapps.
   * **CreateYappScreen**: Provides tools to create Yapps.
4. **Widgets (yapp\_list\_item.dart)**:
   * Displays individual Yapp items with actions like play and share.

**Dependencies**

* provider: State management.
* ffmpeg\_kit\_flutter: Audio/video processing.
* video\_player: Video playback.
* permission\_handler: Handles runtime permissions.
* image\_picker: Captures/selects images.
* record: Records audio.

**Design**

* **Main.dart** initializes the application with the YappProvider.
* **YappProvider** maintains the Yapp list and state changes.
* **CreateYappScreen** processes media files using FFmpeg and saves the output.

**Development Report**

**Challenges and Solutions**

**1. Media File Integration**

* **Challenge**: Combining image and audio into a video using FFmpeg.
* **Solution**: Utilized the ffmpeg\_kit\_flutter package with carefully crafted FFmpeg commands.

**2. Audio Recording Limitations**

* **Challenge**: Enforcing a 2-minute recording limit.
* **Solution**: Implemented a timer with automatic stop functionality.

**3. Permission Management**

* **Challenge**: Handling runtime permissions for storage, camera, and audio.
* **Solution**: Used the permission\_handler package to request and verify permissions.

**4. State Management**

* **Challenge**: Synchronizing changes across screens.
* **Solution**: Used the provider package for efficient state updates.

**5. Responsive UI**

* **Challenge**: Adapting layouts for different screen sizes.
* **Solution**: Used responsive widgets and layouts with constraints.

**Future Enhancements**

* Add support for multiple audio tracks.
* Enhance video playback options with more controls.
* Introduce cloud storage for Yapps.