ARTWORK AUCTIONING PLATFORM WITH AI ASSISTED PRICE GENERATION

A MINI PROJECT REPORT

AI19511- MOBILE APPLICATION DEVELOPMENT LABORATORY FOR ML AND DL APPLICATIONS

Submitted By

HARISH S (221501039)

KISHORE V G (221501062)

Under The Guidance Of

Dr.R. Pavithra Guru

In partial fulfillment for the award of the degree of

BACHELOR OF TECHNOLOGY

In

ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING

RAJALAKSHMI ENGINEERING COLLEGE

An AUTONOMOUS Institution

Affiliated to ANNA UNIVERSITY, Chennai

RAJALAKSHMI ENGINEERING COLLEGE KANCHIPURAM, ANNA UNIVERSITY CHENNAI- 600 025

RAJALAKSHMI
ENGINEERING COLLEGE

BONAFIDE CERTIFICATE

NAME …………………………………………………………………………………..……..…

ACADEMIC YEAR………………..………SEMESTER…………BRANCH………………

UNIVERSITY REGISTER No.

Certified that this is the bonafide record of work done by the above students in the Mini Project titled **"ARTWORK AUCTIONING PLATFORM WITH AI ASSISTED PRICE GENERATION"** in the subject **AI19511 – MOBILE APPLICATION DEVELOPMENT** during the year **2024 - 2025.**

Signature of Faculty – in – Charge

Submitted for the Practical Examination held on _____________________

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ACKNOLEDGEMENT

Initially I thank the Almighty for being with us through every walk of my life and showering his blessings through the endeavor to put forth this report.

My sincere thanks to our Chairman **Mr. S. MEGANATHAN, M.E., F.I.E.,** and our Chairperson **Dr. (Mrs.)THANGAMMEGANATHAN,M.E.,Ph.D.,** for providing me with the requisite infrastructure and sincere endeavoring educating me in their premier institution.

My sincere thanks to **Dr. S.N. MURUGESAN, M.E., Ph.D.,** our beloved Principal for his kind support and facilities provided to complete our work in time.

I express my sincere thanks to **Dr. K.SEKAR, M.E., Ph.D.,** Head of the Department of Artificial Intelligence and Machine Learning and Artificial Intelligence and Data Science for his guidance and encouragement throughout the project work. I convey my sincere and deepest gratitude to our internal guide, **Dr. PAVITHRA GURU (Ph.D).,** Associate Professor, Department of Artificial Intelligence and Machine Learning, Rajalakshmi Engineering College for his valuable guidance throughout the course of the project.

Finally I express my gratitude to my parents and classmates for their moral support and valuable suggestions during the course of the project.

ABSTRACT

This project focuses on developing a mobile application platform dedicated to buying, selling, and auctioning artworks. The platform aims to provide a seamless user experience for artists, collectors, and enthusiasts, offering a marketplace that caters to diverse artistic needs. A standout feature of this platform is an AI-assisted price generation system, designed to empower artists with data-driven pricing suggestions for their artworks. The AI model analyzes multiple factors, including the geographical location of the artist, and comparative prices of similar artworks in the same category. This ensures fair pricing, improves market transparency, and enhances user confidence in transactions. The platform also integrates auction functionality, enabling users to auction their artworks in a competitive environment. A user-friendly interface allows easy uploading of artworks, detailed categorization, and personalized recommendations for buyers. By leveraging artificial intelligence and modern technology, this project not only democratizes access to art markets but also fosters a sustainable ecosystem for artists and collectors. It aims to bridge the gap between creators and consumers, making art more accessible and its trade more efficient.

.

Keywords: *flutter, dart, art marketplace, artwork auction platform, AI-powered pricing, geographical market analysis, competitive bidding, fair art pricing.*

TABLE OF CONTENTS

CHAPTER 1
INTRODUCTION

The art market has evolved significantly with the advent of digital platforms, enabling artists, collectors, and enthusiasts to connect in new and innovative ways. Traditional methods of buying, selling, and auctioning artworks often involve intermediaries, limited accessibility, and challenges in determining fair market value. This project introduces a mobile app platform designed to address these challenges by providing a comprehensive space for buying, selling, and auctioning artworks.

A key feature of this platform is its AI-assisted price generation system. Pricing artworks can be a subjective and often confusing process. To simplify this process, the AI model analyzes data such as similar artworks' prices and geographical trends to suggest an optimal price for each artwork. This empowers artists, especially emerging creators, to price their work competitively and fairly, while also helping buyers make informed purchasing decisions.

In addition to pricing assistance, the platform also facilitates artwork auctions, allowing users to auction their creations in a competitive environment, further driving market engagement. With user-friendly features like secure payment gateways, real-time bidding updates, and personalized recommendations, the app fosters a safe and dynamic marketplace for art transactions.

By leveraging advanced AI technology and creating a seamless user experience, this platform seeks to revolutionize the way art is bought, sold, and valued. It provides a space for artists to showcase their work to a global audience, while also enabling collectors and buyers to make informed decisions. Ultimately, the platform aims to bridge the gap between creators and consumers, making the art world more accessible and its trade more efficient.

CHAPTER 2
LITERATURE REVIEW

1. Analysis of Cross Platform Application Development Over Multiple Devices using Flutter & Dart (Shivam Jadaun, 2023)

The paper focuses on the benefits of using cross-platform application development with Flutter. The authors highlight a key challenge in traditional app development, where separate codebases are needed for each operating system (Android and iOS). To address this, the authors propose using Dart programming language with Flutter, enabling developers to write one codebase that works across multiple operating systems, thus simplifying and streamlining the development process.

2. Impact of Flutter Technology in Software Development (Naveen Kumar Gupta, 2025)

The authors highlight several key features of Flutter, including its hot reload capability, which allows developers to instantly preview changes in real time without restarting the application. This feature significantly accelerates the development process. Additionally, Flutter's declarative UI programming model and highly customizable widgets enable developers to create visually appealing and responsive interfaces. They also note that Flutter apps are compiled directly to native machine code, ensuring smooth performance and animations.

3. E-commerce Smartphone Application (Robert Goodwin, 2012)

In this paper, the authors explore the evolving landscape of mobile and e-commerce applications, emphasizing their role in facilitating online access and the purchase of products and services. The authors highlight the rapid technological advancements driving the evolution of these applications. The paper provides an analysis of the requirements for mobile devices and websites designed for mobile commerce (m-commerce), addressing key aspects such as device specifications, design, and security considerations.

4. **Auctioneer strategy and pricing: evidence from an art auction (Clare D'Souza, 2002)**

This study examines the role of the auctioneer’s strategy used in determining pricing at art auctions. It characterizes the role of auctioneers, prices at which the paintings were sold and tests a series of hypotheses about their behavior. This is done using pricing data from an auction of paintings. It examines the relationship between the auctioneer’s estimates and realized prices. It determines whether the auctioneer and the market evaluate different attributes of different paintings differently and, finally, an analysis is undertaken on the determinants of prices in the art market as suggested by hedonic regression.

5. **Linear regression analysis study (Khushbu Kumari, 2018)**

Linear regression is a statistical procedure for calculating the value of a dependent variable from an independent variable. Linear regression measures the association between two variables. It is a modeling technique where a dependent variable is predicted based on one or more independent variables. Linear regression analysis is the most widely used of all statistical techniques. This article explains the basic concepts and explains how we can do linear regression calculations in SPSS and excel.

2.1 Artwork Auctioning Platforms

- **Online Auction Marketplaces**

  Online auction platforms like eBay, Christie’s, and Artsy have transformed traditional art auctions. They emphasize user-friendly interfaces, scalability, and secure payment mechanisms.

  - *Reference:* Singh et al., "The Role of Online Auctions in the Art Market" (2022).
  - Insights: Platforms need advanced features like fraud detection, secure transactions, and real-time bidding.

- **Blockchain in Auctions**
  Blockchain ensures secure and transparent ownership records for artworks, solving issues of provenance and counterfeiting.
  - *Reference:* Nakamoto et al., "Blockchain for Provenance in Digital Art Auctions" (2021).
  - Insights: Incorporating blockchain can add trust and credibility to the platform.

2.2 AI-Assisted Price Generation

- **Machine Learning for Price Prediction**
  Predicting artwork prices involves analyzing historical data, artist reputation, artwork medium, dimensions, and market trends.
  - *Reference:* Zhang et al., "Predicting Artwork Prices Using Machine Learning Models" (2020).
  - Methods: Regression algorithms (Linear Regression, Random Forest) and neural networks are commonly used for price prediction.
- **Deep Learning in Artwork Valuation**
  Deep learning models like Convolutional Neural Networks (CNNs) can analyze

visual features such as style, color, and complexity to assess an artwork’s value.

  - *Reference:* Chen et al., "Deep Learning Models for Visual Feature-Based Art Price Estimation" (2021).
  - Insights: Models trained on extensive datasets like WikiArt achieve higher accuracy in price predictions.

- **Factors Influencing Art Prices**
  Studies show that art prices depend on a combination of subjective (artistic appeal) and objective (artist reputation, exhibition history) factors.
  - *Reference:* Throsby et al., "Determinants of Art Prices in Auction Markets" (2019).

2.3 Challenges in Art Valuation

- **Subjectivity of Art**
  Valuation is challenging due to the subjectivity involved in artistic appreciation.
  - *Reference:* Smith et al., "Subjectivity in Art Valuation: The Role of AI" (2020).
  - Solutions: AI models must incorporate both structured (e.g., historical sales data) and unstructured data (e.g., expert reviews).
- **Data Scarcity and Quality**
  Art sales data is often limited and non-uniform. Building high-quality datasets is essential for robust AI predictions.
  - *Reference:* Gupta et al., "Data Challenges in AI-Powered Auction Systems" (2021).

2.4 Related Technologies

- **Natural Language Processing (NLP) for Descriptions**
  NLP can analyze artwork descriptions and reviews to determine emotional and thematic value.
  - *Reference:* Lee et al., "Text Mining for Art Description Analysis in Auctions" (2022).
- **Augmented Reality (AR) Integration**
  AR technologies help buyers visualize artworks in their spaces, enhancing purchase confidence.
  - *Reference:* Roy et al., "Enhancing User Experience with AR in Art Auctions" (2023).

3. Gaps Identified

1. Lack of publicly available, large-scale datasets for training AI models on art pricing.
2. Limited research on integrating visual and textual data for holistic art valuation.
3. Underutilization of blockchain for ensuring trust and transparency in online art auctions.

4. Proposed Contributions

- Develop a hybrid AI model that integrates structured and unstructured data for pricing.
- Leverage blockchain to authenticate artwork provenance and ensure secure transactions.
- Implement an AR module for enhancing user engagement and visualization.

CHAPTER 3
PROPOSED SOLUTION

Develop an **AI-powered artwork auctioning platform** that integrates advanced technologies to address these challenges:

1. **AI-Assisted Price Generation:** Use machine learning models to predict the price of artworks based on historical sales, artist reputation, and visual/textual features.
2. **Blockchain for Provenance:** Implement blockchain to record the history and authenticity of artworks, ensuring trust and transparency.
3. **Global Accessibility:** Design a user-friendly web and mobile platform accessible to artists, buyers, and collectors worldwide.
4. **Augmented Reality (AR):** Allow users to visualize artworks in their spaces before purchasing to enhance buyer confidence.
5. **Market Insights and Recommendations:** Provide data-driven insights and recommendations to users based on trends, preferences, and AI analysis.

CHAPTER 4

Methodology

4.1 Data Collection

- **Source Data:**
  - Collect historical sales data from auction houses, online art marketplaces (e.g., Artsy, Sotheby's), and publicly available databases.
  - Scrape visual features and metadata (e.g., size, medium, artist reputation) from platforms like WikiArt.
  - Gather textual data (artwork descriptions, reviews, and critiques) using web scraping tools.
- **Data Preprocessing:**
  - Normalize and clean data to handle missing or inconsistent values.
  - Label data to identify features like style, medium, and time period.

4.2 AI-Assisted Price Prediction Model

- **Features:**
  - **Structured Data:** Artwork dimensions, medium, year of creation, artist reputation, historical prices.
  - **Unstructured Data:**
    - **Visual Features:** Extracted using Convolutional Neural Networks (e.g., ResNet or VGG).
    - **Textual Features:** Processed using Natural Language Processing (NLP) techniques (e.g., BERT).
- **Model Architecture:**

    - Combine structured data with embeddings from visual and textual models using a hybrid approach.
    - Use regression algorithms (e.g., Random Forest Regressor, XGBoost) or neural networks for price prediction.
- **Evaluation Metrics:**
    - Mean Absolute Error (MAE), Mean Squared Error (MSE), and R^2 to evaluate model performance.

4.3 Blockchain Integration

- Use Ethereum or other blockchain frameworks to record:
    - Artwork provenance (ownership history, creation details).
    - Transaction records to ensure transparency and prevent fraud.
- Develop smart contracts for:
    - Automatic bidding and payment handling.
    - Transferring ownership upon successful bids.

4.4 Platform Development

- **Frontend:**
    - Develop a responsive web and mobile application using frameworks like Flutter or React.
    - Integrate an Augmented Reality (AR) module for artwork visualization.
- **Backend:**
    - Use FastAPI or Django for the backend to handle user management, auctions, and AI model integration.
    - Store and manage data in a cloud database (e.g., AWS RDS, Firebase).

4.5 Augmented Reality (AR) Module

- Use AR SDKs (e.g., ARKit for iOS, ARCore for Android) to enable users to:
    - Place artworks in virtual spaces for real-time visualization.
    - Adjust scale and lighting for realistic previews.

4.6 User Features

1. **For Artists:**
    - List artworks with descriptions, images, and metadata.
    - AI-assisted pricing suggestions for new listings.
2. **For Buyers:**
    - Browse artworks by category, artist, or price range.
    - Access AI-predicted price ranges and visualizations.
3. **For Collectors:**
    - Analyze market trends and historical price trajectories.
    - Track and manage owned artworks with blockchain-based proof of ownership.

4.7 Testing and Deployment

- **Testing:**
    - Unit testing for individual modules (e.g., AI model, blockchain, AR).
    - Integration testing to ensure seamless interaction between components.
    - Usability testing with end-users (artists, buyers, collectors).
- **Deployment:**
    - Deploy the platform on a cloud service (e.g., AWS, Azure).
    - Use Docker for containerization and Kubernetes for scaling.

4.8 Maintenance and Scalability

- Continuously update AI models with new data to improve accuracy.
- Monitor user feedback and platform analytics to optimize performance.
- Implement caching and load balancing for handling high traffic.

CHAPTER 5
SYSTEM REQUIREMENTS

5.1 Hardware Requirements

1. **Processor (CPU)**:
   - Minimum: Quad-core processor (or equivalent)
   - Recommended: Octa-core processor for better performance
2. **RAM**:
   - Minimum: 2 GB RAM
   - Recommended: 4 GB or more for better multitasking and running more resource-intensive apps
3. **Storage**:
   - Minimum: 100 MB of available storage
   - Recommended: 500 MB or more of free storage for optimal performance

5.2 Software Requirements

1. **Operating System (OS)**:
   - **Android**:
     - Minimum: Android 5.0 (Lollipop) or later
   - **iOS**:
     - Minimum: iOS 11.0 or later
2. **Internet Connection** (if applicable):
   - Required for apps that rely on cloud services, data syncing, or other online features. A stable 3G/4G/5G or Wi-Fi connection is recommended for seamless operation.

CHAPTER 6

SYSTEM OVERVIEW

6.1 EXISTING SYSTEM

Current systems for art buying, selling, and auctioning primarily consist of traditional online marketplaces, auction platforms, and valuation services:

1. **Online Art Marketplaces**:
   Platforms like Etsy and Saatchi Art allow artists to list their works for sale. While these platforms offer a space for artists to display their artworks, pricing is often subjective, relying on artists' personal judgment or comparable listings without advanced tools to assist in price determination.
2. **Auction Platforms**:
   Websites like eBay and Paddle8 facilitate art auctions, but they primarily cater to high-end artworks and have limited geographic reach. These platforms lack personalized pricing recommendations and often fail to provide real-time market insights or localized trends.
3. **Art Valuation Services**:
   Traditional art appraisal services offer pricing based on expert evaluation and historical data but are typically costly and time-consuming. They don’t provide instant or AI-driven insights into current market conditions.

6.1.1 DRAWBACKS OF EXISTING SYSTEM

Absence of AI-Assisted Pricing:
Existing systems lack AI tools to analyze market trends, artist history, and location-based factors to suggest optimal pricing for artworks.

Limited Personalization:

Generic pricing strategies do not account for individual artist profiles or geographic demand variations, hindering effective pricing decisions.

Geographical Limitations:

Current systems fail to factor in location-specific influences on art pricing, leading to a one-size-fits-all approach.

6.2 PROPOSED SYSTEM

The proposed system combines a mobile platform for buying, selling, and auctioning artwork with AI-powered price recommendations tailored to market trends, artist history, and geographic factors.

1. AI-Assisted Price Recommendation

- **Data-Driven Pricing:** AI analyzes historical pricing, artist sales, and regional trends to recommend fair prices for artworks.
- **Geographic Influence:** Price suggestions will consider location-based demand, ensuring relevance to local art markets.
- **Dynamic Adjustments:** Prices will update in real-time, reflecting market conditions and trends.

2. Art Marketplace

- **User-Friendly Interface:** Artists can upload and manage their artworks, while buyers can filter artworks by style, artist, and price.
- **Secure Transactions:** Multiple payment options will be available, ensuring secure purchases.
- **Personalized Recommendations:** Buyers receive art suggestions based on preferences and local trends.

3. Auction Platform

- **Real-Time Bidding:** Users can participate in live auctions, receiving updates and notifications during bidding.
- **Auction Analytics:** Sellers and buyers can track trends and adjust their strategies in real-time.

CHAPTER 7

SYSTEM IMPLEMENTATION

7.1 SYSTEM ARCHITECTURE

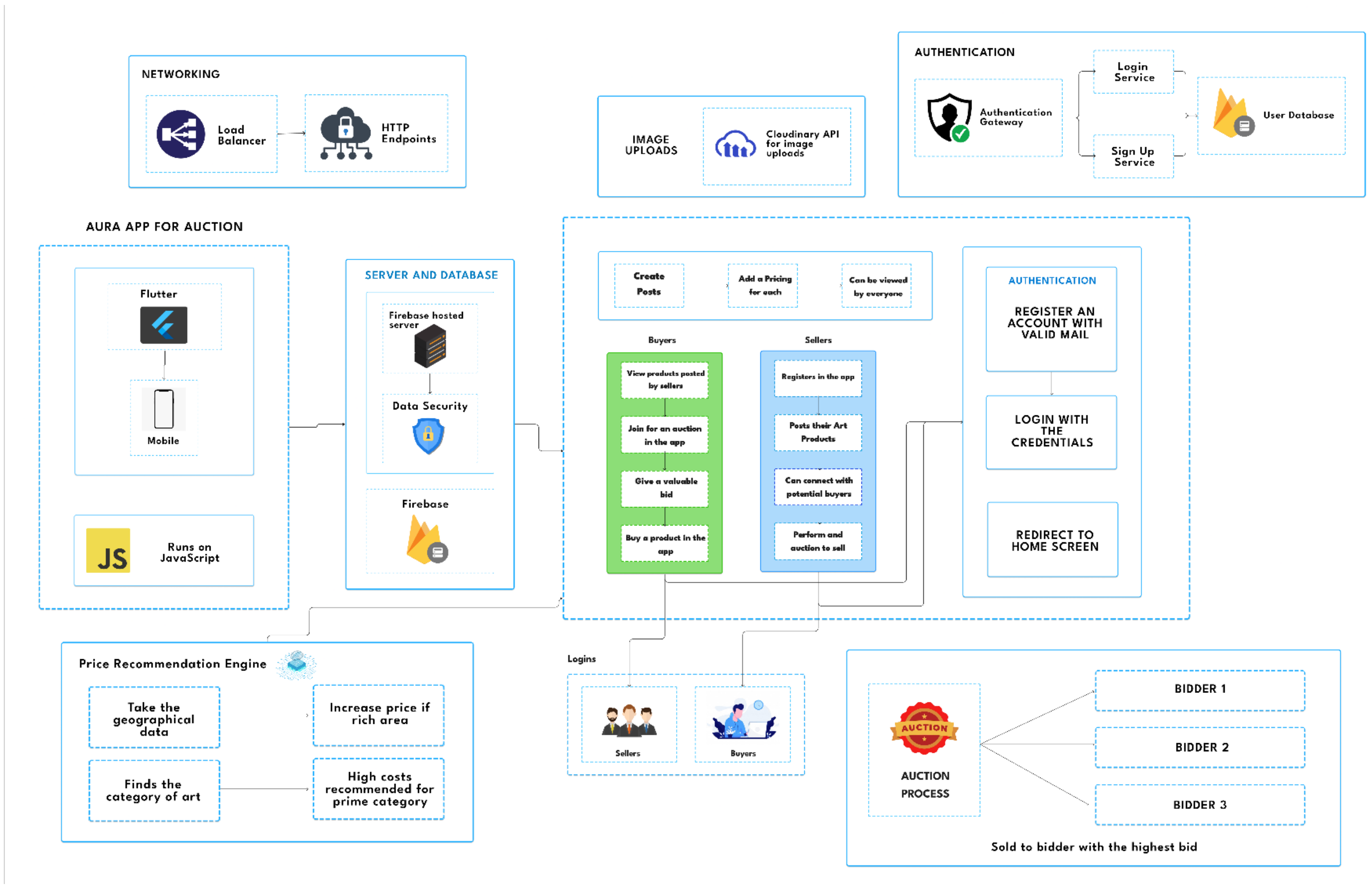


Fig 7.1 Architecture diagram

The user interacts with the Flutter app on their mobile device. The app communicates with the Firebase backend to handle authentication, image uploads, and data storage. The price recommendation engine calculates optimal prices based on item category and historical data. The auction process is managed on the server, with the highest bidder winning the item.

7.2 SYSTEM FLOW

The system flow begins with User Registration & Login, allowing users to sign up as artists or buyers. Artists then Upload Artwork and receive AI-generated price recommendations. Buyers browse the Art Marketplace to explore artworks, while artists can list pieces for auction in the Auction Platform. Buyers place bids in real-time, and once a purchase is made, the buyer proceeds to Checkout for secure payment and transaction confirmation. This process ensures a seamless experience for both artists and buyers.

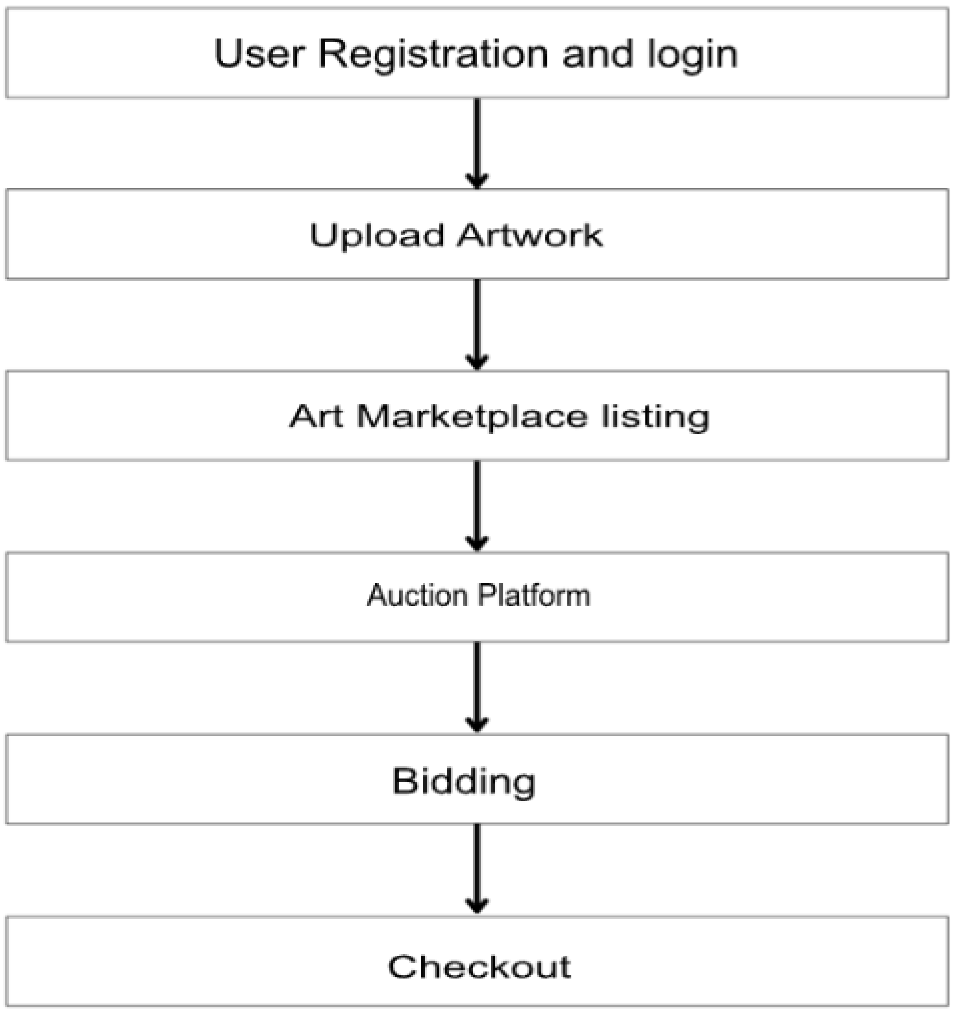


Fig 7.2 Workflow diagram

7.3 LIST OF MODULES

1. UI/UX Design
2. Frontend Development
3. Backend Development
4. Model Training
5. Model Integration
6. APK Building

7.4 MODULE DESCRIPTION

I. UI/UX Design:

The UI/UX design process focused on creating an intuitive and visually appealing interface for the mobile application. Emphasis was placed on easy navigation and a seamless user experience, ensuring that the app is both user-friendly and engaging. The layout and design elements were carefully crafted to enhance user satisfaction, with a focus on clarity and aesthetics, which contributes to improved user retention and overall interaction.

II. Frontend Development:

Frontend development involved using Flutter to implement the user interface, ensuring that the app's design was translated into a fully functional and responsive application. This included building dynamic layouts, handling animations, and ensuring compatibility across various devices. The frontend was integrated with backend services to ensure smooth data flow, delivering a fast, interactive, and seamless experience for users across all platforms.

III. Backend Development:

Backend development focused on creating the server-side architecture necessary to support the mobile application. This involved setting up databases, configuring APIs, and handling data management, secure transactions, and user authentication. The backend ensures that data requests are processed efficiently and securely, facilitating real-time communication between the frontend and external systems to guarantee the smooth and reliable operation of the application.

IV. Model Training:

The model training process involved training an AI model to accurately recommend prices for artworks. By analyzing historical artwork sales, regional trends, and other relevant data, the model learned to predict optimal pricing for artworks. Continuous training with new data improved the accuracy of the model, making it more reliable in providing price recommendations for artists and buyers.

V. Model Integration:

Model integration ensured that the trained AI model was embedded into the application’s backend and seamlessly interacted with the user interface. This allowed the model to generate real-time price recommendations based on user input. Integration also involved ensuring that the backend could process and deliver predictions efficiently, ensuring the AI's predictions remained accurate and that users received a smooth experience when uploading and pricing artworks.

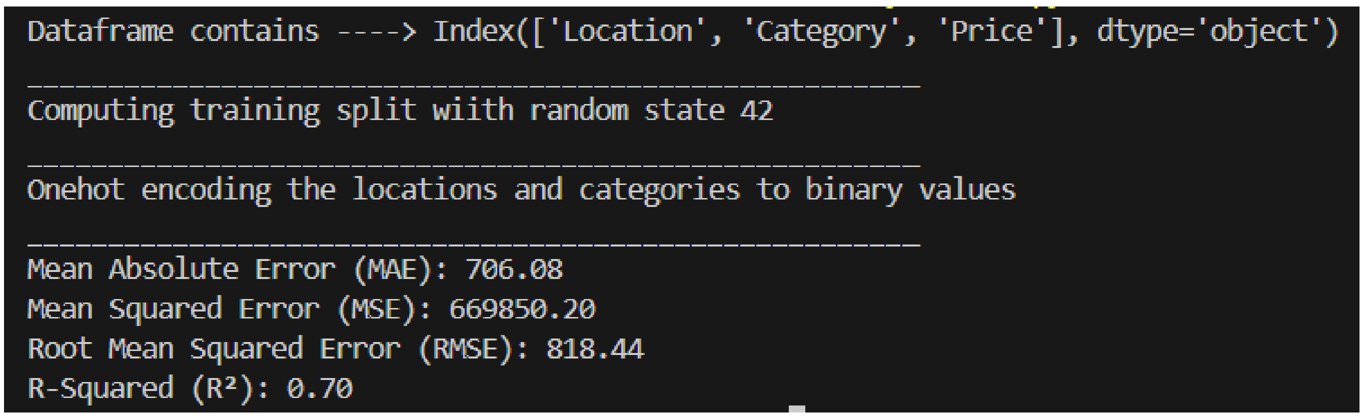
VI. APK Building:

The APK building process involved converting the Flutter app’s code into an Android Package (APK) file for distribution. This ensured the app was optimized for Android devices, addressing performance, compatibility, and security concerns. The APK was tested for functionality on various devices and finalized for deployment to the Google Play Store, allowing the app to be installed and run seamlessly across a range of Android devices.

CHAPTER-8

RESULT AND DISCUSSION

The mobile application prototype achieved its key objectives, successfully demonstrating the core functionalities of buying, selling, and auctioning artworks, alongside AI-assisted price recommendations. The user interface was intuitive and responsive, with navigation and design elements aligning well with the overall user experience goals. Early feedback from internal testing suggested that the design was visually appealing and easy to navigate, providing a positive initial impression.



```
Dataframe contains ----> Index(['Location', 'Category', 'Price'], dtype='object')
_______________________________________________________
Computing training split wiith random state 42
_______________________________________________________
Onehot encoding the locations and categories to binary values
_______________________________________________________
Mean Absolute Error (MAE): 706.08
Mean Squared Error (MSE): 669850.20
Root Mean Squared Error (RMSE): 818.44
R-Squared (R²): 0.70
```

Fig 6.1 Price generation model results

The AI model, trained to recommend artwork prices based on historical sales and geographical data, showed promising results during the development phase. Price predictions generated by the model were within an acceptable range when compared to expected market values, indicating the model's capability to assist artists in determining appropriate pricing for their artworks.

In conclusion, the system successfully met the expectations for both functionality and user experience. With some optimizations, particularly in server performance and model training, the platform has the potential to offer a highly effective solution for the artwork buying and selling community.

APPENDIX

SAMPLE CODE

main.dart

```
import 'package:provider/provider.dart';
import 'package:flutter/material.dart';

import 'package:flutter_localizations/flutter_localizations.dart';
import 'package:flutter_web_plugins/url_strategy.dart';
import 'auth/firebase_auth/firebase_user_provider.dart';
import 'auth/firebase_auth/auth_util.dart';

import 'backend/firebase/firebase_config.dart';
import '/flutter_flow/flutter_flow_theme.dart';
import 'flutter_flow/flutter_flow_util.dart';
import 'index.dart';

void main() async {
  WidgetsFlutterBinding.ensureInitialized();
  GoRouter.optionURLReflectsImperativeAPIs = true;
  usePathUrlStrategy();

  await initFirebase();

  final appState = FFAppState(); // Initialize FFAppState
  await appState.initializePersistedState();

  runApp(ChangeNotifierProvider(
    create: (context) => appState,
    child: const MyApp(),
  ));
}

class MyApp extends StatefulWidget {
  const MyApp({super.key});

  // This widget is the root of your application.
  @override
  State<MyApp> createState() => _MyAppState();
  static _MyAppState of(BuildContext context) =>
      context.findAncestorStateOfType<_MyAppState>()!;
}

class _MyAppState extends State<MyApp> {
  ThemeMode _themeMode = ThemeMode.system;

  late AppStateNotifier _appStateNotifier;
  late GoRouter _router;

  late Stream<BaseAuthUser> userStream;

  @override
  void initState() {
    super.initState();

    _appStateNotifier = AppStateNotifier.instance;
    _router = createRouter(_appStateNotifier);
    userStream = referenceaaFirebaseUserStream()
      ..listen((user) {
        _appStateNotifier.update(user);
      });
    jwtTokenStream.listen((_) {});
    Future.delayed(
      const Duration(milliseconds: 1000),
      () => _appStateNotifier.stopShowingSplashImage(),
    );
  }

  void setThemeMode(ThemeMode mode) => safeSetState(() {
        _themeMode = mode;
      });

  @override
  Widget build(BuildContext context) {
    return MaterialApp.router(
      title: 'referenceaa',
      localizationsDelegates: const [
        GlobalMaterialLocalizations.delegate,
        GlobalWidgetsLocalizations.delegate,
        GlobalCupertinoLocalizations.delegate,
      ],
      supportedLocales: const [Locale('en', '')],
      theme: ThemeData(
        brightness: Brightness.light,
        useMaterial3: false,
      ),
      themeMode: _themeMode,
      routerConfig: _router,
    );
  }
}

class NavBarPage extends StatefulWidget {
  const NavBarPage({super.key, this.initialPage, this.page});

  final String? initialPage;
  final Widget? page;

  @override
  _NavBarPageState createState() => _NavBarPageState();
}

/// This is the private State class that goes with NavBarPage.
class _NavBarPageState extends State<NavBarPage> {
  String _currentPageName = 'Home';
  late Widget? _currentPage;

  @override
  void initState() {
    super.initState();
    _currentPageName = widget.initialPage ?? _currentPageName;
    _currentPage = widget.page;
  }

  @override
  Widget build(BuildContext context) {
    final tabs = {
      'Home': const HomeWidget(),
      'Cart': const CartWidget(),
      'favorite': const FavoriteWidget(),
      'auctionForm': const AuctionFormWidget(),
    };
```

```
final currentIndex = tabs.keys.toList().indexOf(_currentPageName);

return Scaffold(
  body: _currentPage ?? tabs[_currentPageName],
  bottomNavigationBar: BottomNavigationBar(
    currentIndex: currentIndex,
    onTap: (i) => safeSetState(() {
      _currentPage = null;
      _currentPageName = tabs.keys.toList()[i];
    }),
    backgroundColor: Colors.white,
    selectedItemColor: FlutterFlowTheme.of(context).primary,
    unselectedItemColor: const Color(0x8A000000),
    showSelectedLabels: false,
    showUnselectedLabels: false,
    type: BottomNavigationBarType.fixed,
    items: const <BottomNavigationBarItem>[
      BottomNavigationBarItem(
        icon: Icon(
          Icons.home_rounded,
          size: 24.0,
        ),
        label: 'Home',
        tooltip: '',
      ),
      BottomNavigationBarItem(
        icon: Icon(
          Icons.shopping_cart,
          size: 24.0,
        ),
        label: 'Cart',
        tooltip: '',
      ),
      BottomNavigationBarItem(
        icon: Icon(
          Icons.favorite,
          size: 24.0,
        ),
        label: 'Favorites',
        tooltip: '',
      ),
      BottomNavigationBarItem(
          icon: Icon(
            Icons.sell_rounded,
            size: 24.0,
          ),
          label: 'Submit',
          tooltip: '',
        )
      ],
    ),
  );
 }
}
```

Pubspec.yaml

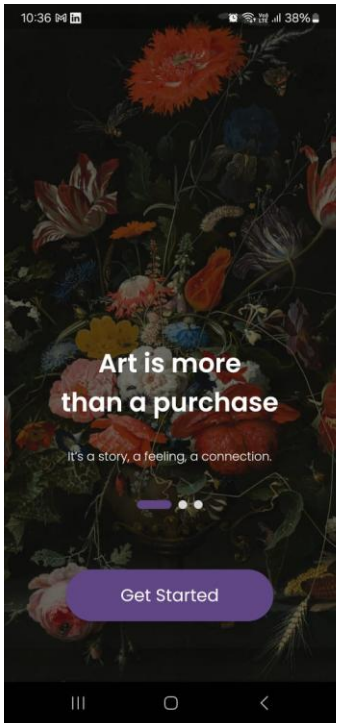
```
name: Aura
description: A new Flutter project.

# The following line prevents the package from being accidentally published to
# pub.dev using `pub publish`. This is preferred for private packages.
publish_to: "none" # Remove this line if you wish to publish to pub.dev

# The following defines the version and build number for your application.
# A version number is three numbers separated by dots, like 1.2.43
# followed by an optional build number separated by a +.
# Both the version and the builder number may be overridden in flutter
# build by specifying --build-name and --build-number, respectively.
# In Android, build-name is used as versionName while build-number used as
versionCode.
# Read more about Android versioning at
https://developer.android.com/studio/publish/versioning
# In iOS, build-name is used as CFBundleShortVersionString while build-number used as
CFBundleVersion.
# Read more about iOS versioning at
#
https://developer.apple.com/library/archive/documentation/General/Reference/InfoPlistKe
yReference/Articles/CoreFoundationKeys.html
version: 1.0.0+1

environment:
  sdk: ">=3.0.0 <4.0.0"
```

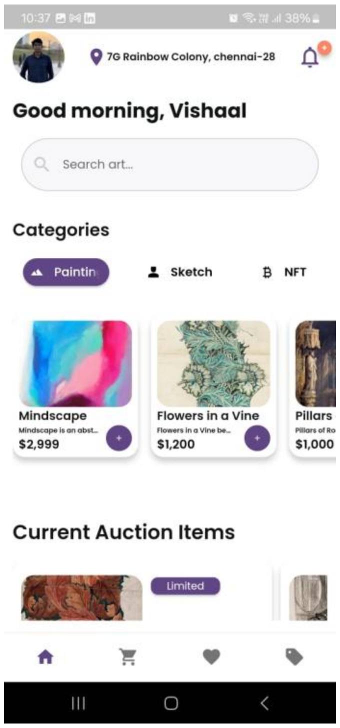
```
dependencies:
  flutter:
    sdk: flutter
  flutter_localizations:
    sdk: flutter
  flutter_web_plugins:
    sdk: flutter
  auto_size_text: 3.0.0
  badges: 2.0.2
  cached_network_image: 3.4.1
  cached_network_image_platform_interface: 4.1.1
  cached_network_image_web: 1.3.1
  cloud_firestore: 5.4.4
  cloud_firestore_platform_interface: 6.4.3
  cloud_firestore_web: 4.3.2
  collection: 1.18.0
  csv: 6.0.0
  firebase_auth: 5.3.1
  firebase_auth_platform_interface: 7.4.7
  firebase_auth_web: 5.13.2
  firebase_core: 3.6.0
  firebase_core_platform_interface: 5.3.0
  firebase_core_web: 2.18.1
  firebase_performance: 0.10.0+8
  firebase_performance_platform_interface: 0.1.4+44
  firebase_performance_web: 0.1.7+2
  flutter_animate: 4.5.0
  flutter_cache_manager: 3.4.1
  flutter_secure_storage: 9.2.2
  flutter_secure_storage_linux: 1.2.1
  flutter_secure_storage_macos: 3.1.2
  flutter_secure_storage_platform_interface: 1.1.2
  flutter_secure_storage_web: 1.2.1
  flutter_secure_storage_windows: 3.1.2
  font_awesome_flutter: 10.7.0
  from_css_color: 2.0.0
  go_router: 12.1.3
  google_fonts: 6.1.0
  google_sign_in: 6.2.1
  google_sign_in_android: 6.1.30
  google_sign_in_ios: 5.7.7
```

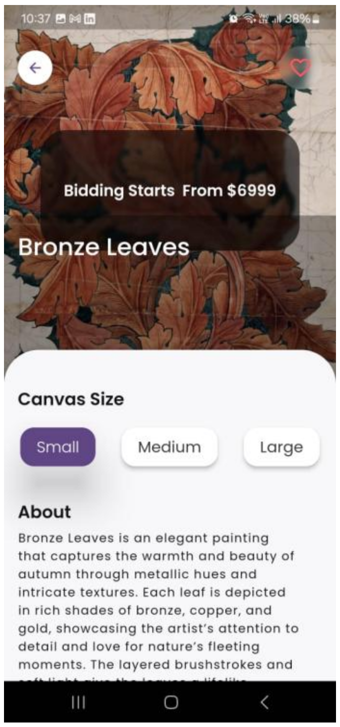
```
google_sign_in_platform_interface: 2.4.5
google_sign_in_web: 0.12.4+2
intl: 0.19.0
json_path: 0.7.2
lottie: 2.7.0
page_transition: 2.1.0
path_provider: 2.1.4
path_provider_android: 2.2.10
path_provider_foundation: 2.4.0
path_provider_linux: 2.2.1
path_provider_platform_interface: 2.1.2
path_provider_windows: 2.2.1
plugin_platform_interface: 2.1.8
provider: 6.1.2
rxdart: 0.27.7
shared_preferences: 2.3.2
shared_preferences_android: 2.3.2
shared_preferences_foundation: 2.5.2
shared_preferences_linux: 2.4.1
shared_preferences_platform_interface: 2.4.1
shared_preferences_web: 2.4.2
shared_preferences_windows: 2.4.1
sign_in_with_apple: 6.1.2
sign_in_with_apple_platform_interface: 1.1.0
sign_in_with_apple_web: 2.1.0
smooth_page_indicator: 1.1.0
sqflite: 2.3.3+1
sqflite_common: 2.5.4+3
stream_transform: 2.1.0
synchronized: 3.2.0
timeago: 3.6.1
url_launcher: 6.3.0
url_launcher_android: 6.3.10
url_launcher_ios: 6.3.1
url_launcher_linux: 3.2.0
url_launcher_macos: 3.2.1
url_launcher_platform_interface: 2.3.2
url_launcher_web: 2.3.3
url_launcher_windows: 3.1.2

# The following adds the Cupertino Icons font to your application.
# Use with the CupertinoIcons class for iOS style icons.
```

```
  cupertino_icons: ^1.0.0

dependency_overrides:
  http: 1.2.2
  uuid: ^4.0.0

dev_dependencies:
  flutter_lints: 4.0.0
  lints: 4.0.0
  flutter_test:
    sdk: flutter

# For information on the generic Dart part of this file, see the
# following page: https://dart.dev/tools/pub/pubspec

# The following section is specific to Flutter.
flutter:
  # The following line ensures that the Material Icons font is
  # included with your application, so that you can use the icons in
  # the material Icons class.
  uses-material-design: true

  # To add assets to your application, add an assets section, like this:
  assets:
    - assets/fonts/
    - assets/images/
    - assets/videos/
    - assets/audios/
    - assets/rive_animations/
    - assets/pdfs/
    - assets/jsons/

  # An image asset can refer to one or more resolution-specific "variants", see
  # https://flutter.dev/assets-and-images/#resolution-aware.

  # For details regarding adding assets from package dependencies, see
  # https://flutter.dev/assets-and-images/#from-packages

  # To add custom fonts to your application, add a fonts section here,
  # in this "flutter" section. Each entry in this list should have a
  # "family" key with the font family name, and a "fonts" key with a
  # list giving the asset and other descriptors for the font. For
# example:
# fonts:
#   - family: Schyler
#     fonts:
#       - asset: fonts/Schyler-Regular.ttf
#       - asset: fonts/Schyler-Italic.ttf
#         style: italic
#   - family: Trajan Pro
#     fonts:
#       - asset: fonts/TrajanPro.ttf
#       - asset: fonts/TrajanPro_Bold.ttf
#         weight: 700
#
# For details regarding fonts from package dependencies,
# see https://flutter.dev/custom-fonts/#from-packages
```

OUTPUT SCREENSHOTS

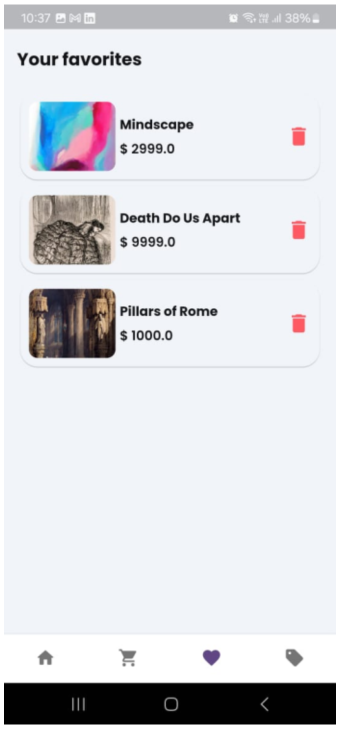

10:36 38%
Art is more than a purchase
It's a story, a feeling, a connection.
Get Started

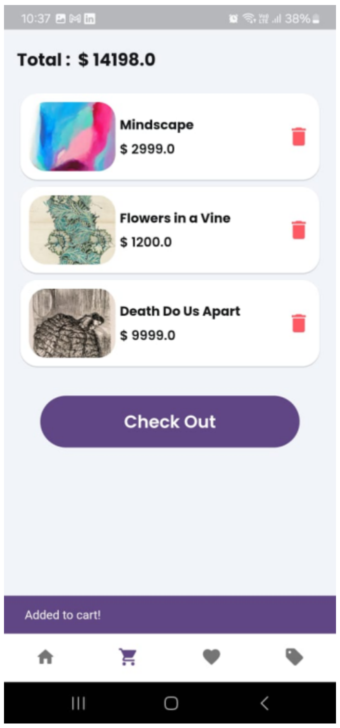




Bronze Leaves is an elegant painting that captures the warmth and beauty of autumn through metallic hues and intricate textures. Each leaf is depicted in rich shades of bronze, copper, and gold, showcasing the artist's attention to detail and love for nature's fleeting moments. The layered brushstrokes and

Fig A.1 Onboarding, Home and Auction Pages





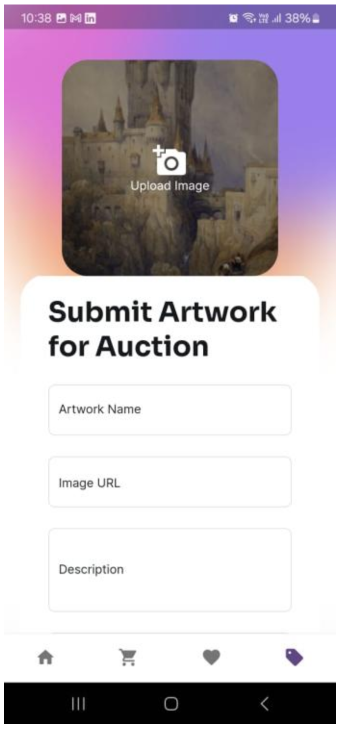


Fig A.2 Favourites, Cart and Art submit Pages

REFERENCE

[1] Jadaun, Shivam & Singh, Rajeev & Kumar, Rohit & Agarwal, Dr. (2023). Analysis of Cross Platform Application Development Over Multiple Devices using Flutter & Dart. International Journal of Recent Technology and Engineering (IJRTE). 12. 33-38. 10.35940/ijrte.A7580.0512123.

[2] Gupta, Naveen. (2024). Impact of Flutter Technology in Software Development. International Journal of Research Publication and Reviews. 5. 3749-3752. 10.55248/gengpi.5.0724.1928.

[3] Alqahtani, Abdullah & Goodwin, Robert. (2012). E-commerce Smartphone Application. International Journal of Advanced Computer Science and Applications. 3. 10.14569/IJACSA.2012.030810.

[4] D'Souza, Clare & Prentice, David. (2002). Auctioneer strategy and pricing: evidence from an art auction. Marketing Intelligence & Planning. 20. 417-427. 10.1108/02634500210450855.

[5] Kumari, Khushbu & Yadav, Suniti. (2018). Linear regression analysis study. Journal of the Practice of Cardiovascular Sciences. 4. 33. 10.4103/jpcs.jpcs_8_18.