Mobile User Experience
6MMCS001W

Coursework 2 Report - Mobile App Implementation

Author: Athindu Umayanga
Student ID: 2018580 | w1742249

Group Number: 18

Links to Web Site: https://lkiea.com/
YouTube URL: https://youtu.be/DiyOmYwhVqs
Code Google Drive Link:
https://drive.google.com/drive/folders/1woLQm1hJtSt14lWtSiFIW9yumlsqSScc?usp=sharing

Content

Table of Figures

2. Design and implementation of mobile web app features

Lkiea mobile web app was implemented based on the previously designed UI-UX design model and a few minor changes were carried out in the implementation to increase the functionality of the system. Mobile web app has used HTML, CSS, jQuery Mobile, JavaScript and Firebase for the development and the developed system was then hosted on a server for faster and easy access.

The system was required to develop on two devices (iPhone, iPad) and in two orientations (Portrait, Landscape) making altogether 4 variants of the system. Hence the welcome page implemented in the hosted server was designed to make it easier to view and access the 4 variants separately without using the inspect option.

Figure 1: Lkiea Home Page

By clicking on the variants you'll be redirected to the Lkiea implementation on the respective device and the orientation.

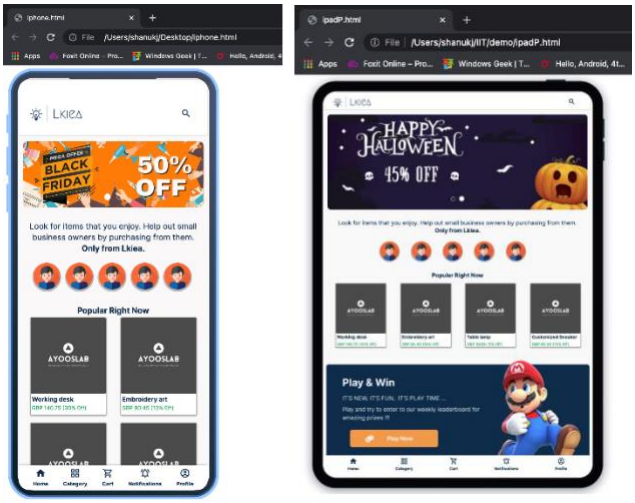


Figure 2: Variants

This report briefly discusses the implementation of the core functionalities app implemented in the Lkiea mobile web (DIY Store) and how the technologies were used to complete each functionality.

2.1 Eye catching introduction

At the initial execution of an application, developers need to consider giving the basic idea about the application to the users, hence the implemented model was integrated with three popup boxes that give the user a gist of the application's purpose. To increase the user experience, users were given the option to view all the introductory popups one by one or skip the popups and directly view the application and the popups were designed in a way so that users won't get the same popup boxes every time they visit the home page of the application.

Figure 3 shows how popups were created and output can be viewed with figure 4.

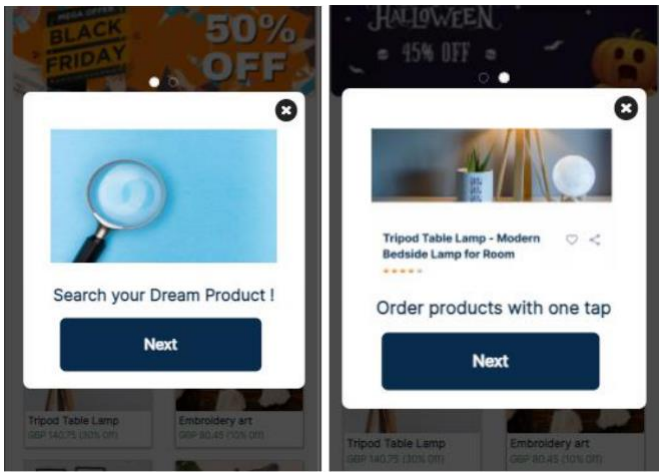


Figure 3: Intro Popups

```
<div class="popUp-home-view" id="popUp-home-intro">
  <div class="overlay-home" onclick="showIntro();"></div>
  <div class="content-home">
    <div class="close-btn-home" onclick="showIntro();">&times;</div>
    <br>
    <div id="swap-first">
      <img
        src="https://res.cloudinary.com/dgly8b9lq/image/upload/v1642264855/Lkia/unsplash_afW1hht0NSs_ihgukz.png"
        alt="" width="306" height="180" class="qr-ar-button" id="popImage" />

      <h4 class="popup-body-home">Search your Dream Product !</h4>
    </div>

    <div id="swap-second" style="display:none;">
      <img
        src="https://res.cloudinary.com/dgly8b9lq/image/upload/v1642888262/Lkia/image_35_q1dmpr.png"
        alt="" width="306" height="180" class="qr-ar-button" id="popImage" />

      <h4 class="popup-body-home">Order products with one tap</h4>
    </div>

    <div id="swap-third" style="display:none;">
      <img
        src="https://res.cloudinary.com/dgly8b9lq/image/upload/v1642888858/Lkia/istockphoto-821114486-170667a_hqdypj.jpg"
        alt="" width="306" height="180" class="qr-ar-button" id="popImage" />

      <h4 class="popup-body-home">Get started</h4>
    </div>

    <a href="javascript:SwapDivsWithClick('swap-first','swap-second','swap-third')">
      <div class="homePop-button" id="homePop-btn">Next</div>
```

Figure 4: Intro Popup implementation

2.2 Incorporation of a short video

A few short videos were incorporated in the Lkiea implementation to give the user a fresh experience about the DIY products that can be found throughout the application. When implementing the short video functionality, social media platform's (WhatsApp, Instagram, etc) statuses or stories idea were used in hope of making the user more comfortable with the application. Incorporated short videos can be considered as some user stories which they have published in the platform after buying their item from the Lkiea platfrom. Five short videos were included in the application and user was given the option to view any story at any time of their choice and user can close the video by just clicking outside of the video frame.

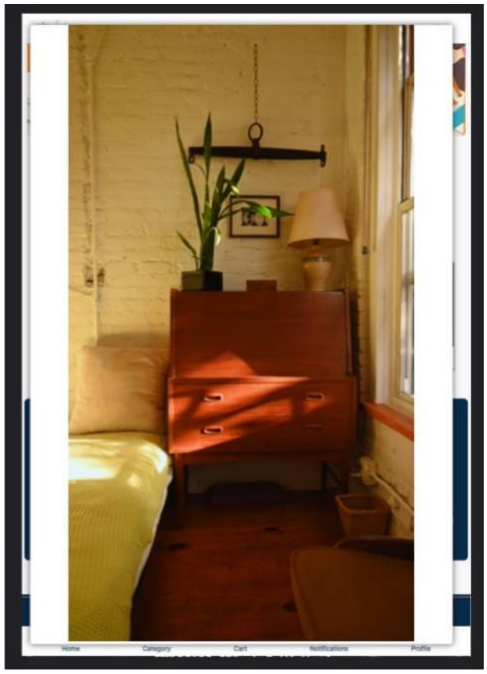


Figure 5: Short Video

The short video was set to autoplay as soon as the user stories were clicked and the video resolution will be adjusted automatically to the relevant screen size as a popup window. Below code was used to implement this functionality and video was used as a popup data role.

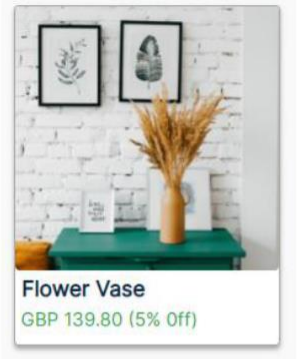
Figure 6: Short Video implementation

```
<div data-role="popup" id="popupVideo">
  <div class="video-container">
    <div class="h_iframe">
      <video controls autoplay loop style="height:calc(100vh - 50px);
        width:calc(100vw - 50px);">
        <source src="https://lkiea.com/mobile-ui-ux/videos/1.mp4" type="video/mp4">
      </video>
    </div>
  </div>
</div>
```

2.3 POI

Several types of POIs are implemented in different parts of the application to draw attention from users to increase the user interactions with the application. Furthermore, POI components are placed in the pages in a way that it's easily accessible and clearly viewed without a hassle.

Product cards can be considered as one of the main points of interest in the application as the system is based on a product-based platform main ambition of the platform is to sell the products and at first glance, users can view the products on a card format before viewing more details of the product by going to the relevant product page. As shown in the below diagram product card includes all the main details (Product Image, Product Name, Price, Sale Percentage) required by the user and the image of the product occupies a large scale of the card as people tend to view images more than text so that it will unintentionally force him to view the product and buy the item.




```
<div class="prod-grid">
  <!-- Card elements -->
  <div class="card_elements">
    <div class="card-image">
      <img alt="home"
        src="https://res.cloudinary.com/dgly8b9lq/image/upload/c_fill,h_500,w_500/v1642916097/Lkia/unsplash_W5dsm9n6e3g_haegd5.png"
        class="image_card" />
    </div>
    <h5 class="card-product-title"> Flower Vase</h5>
    <h6 class="card-product-desc"> GBP 139.80 (5% Off)</h6>
  </div>
</div>
```

Figure 7: POI view and implementation

When screen size changes placement of the product card will also change with the sizes to keep the screen occupied without empty white spaces. The below diagram shows a comparison between card view in iPhone and iPad.

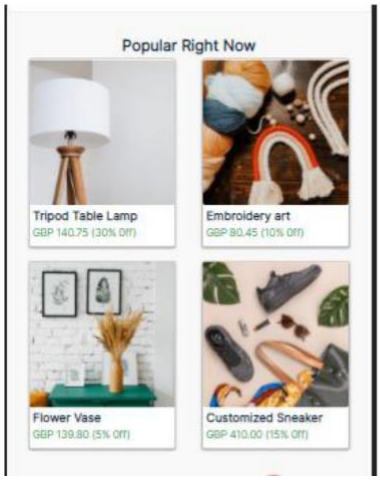


Figure 8: POI comparison iPhone v iPad

2.5 Integration of google maps

A dark-themed google map was used in the implementation as it matches with the designed mobile web application and it was placed just above the footer as a section to show the user where our store/service is based on. As Lkiea is a proposed DIY store around the neighborhood of Westminster (London, England), a marker was placed on that specific area by identifying the relevant longitudes and latitudes of that location. Furthermore, some unnecessary tags were removed from the default map view to show a clear view of the map.

Figure 9: Map Implementation

```
// Initialize and add the map
function initMap() {
  const uluru = { lat: 51.520919, lng: -0.140126 };
  // Styles a map in night mode.
  const map = new google.maps.Map(document.getElementById("map"), {
    center: uluru,
    zoom: 12,
    mapTypeControl: false,
    streetViewControl: false,
    //disableDefaultUI: true,
    styles: [
      { elementType: "geometry", stylers: [{ color: "#242f3e" }] },
      {
        elementType: "labels.text.stroke",
        stylers: [{ color: "#242f3e" }],
      },
      {
        elementType: "labels.text.fill",
        stylers: [{ color: "#746855" }],
      },
      {
        featureType: "administrative.locality",
        elementType: "labels.text.fill",
        stylers: [{ color: "#d59563" }],
      },
      {
        featureType: "poi",
        elementType: "labels.text.fill",
        stylers: [{ color: "#d59563" }],
      },
      {
        featureType: "poi.park",
        elementType: "geometry",
        stylers: [{ color: "#263c3f" }],
      },
```

The map was designed to be responsive for both iPhone and iPad hence map will adjust its view size when device/ orientation changes.

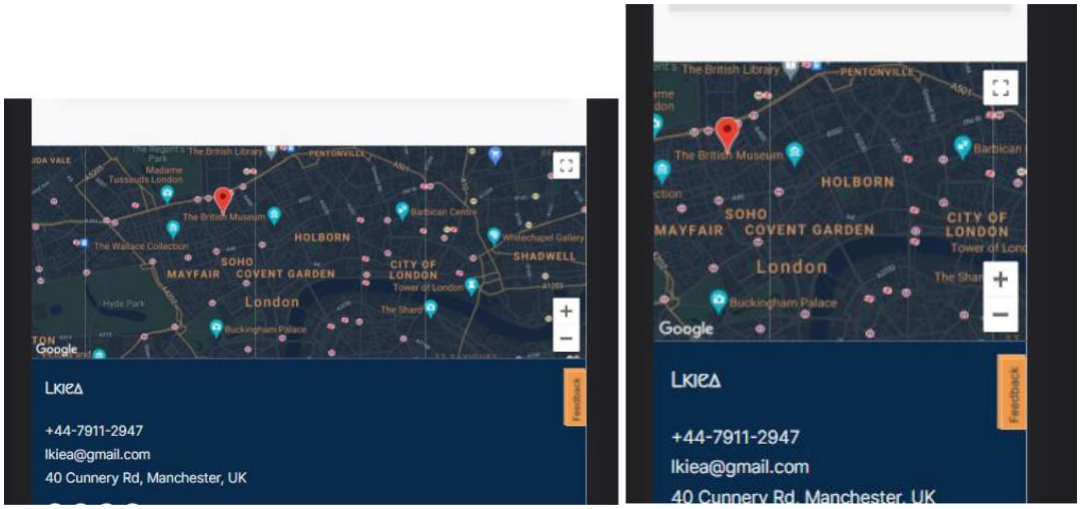
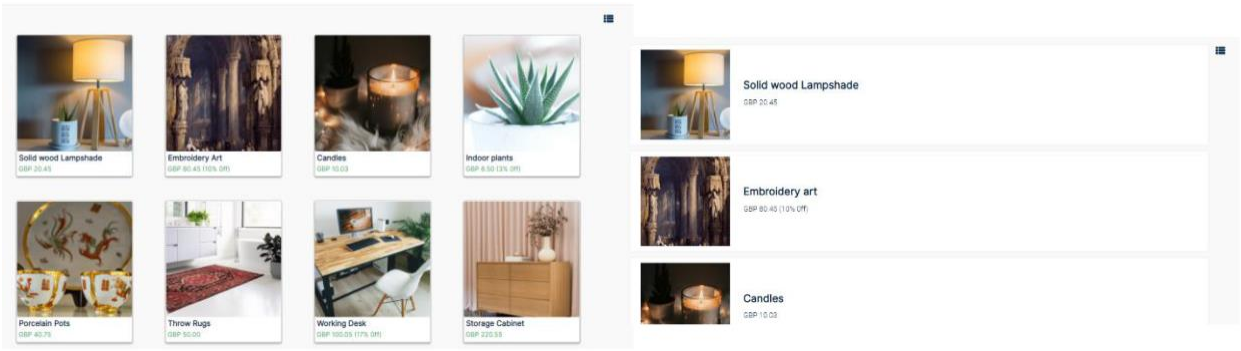


Figure 10: Map View (iPhone v iPad)

2.5 List creation & sorting and filtering widgets

In the default view, the user can view all the available products in a grid format using card view elements, and users were given the option to view the same products in list format using list view elements from a button click.




```
function gridToggle(){
  var l1 = document.getElementById("list_view");
  var g2 = document.getElementById("grid_view");
  l1.classList.toggle("hidden");
  g2.classList.toggle("hidden");
}
```

Figure 11: Grid view , List view

2.6 Booking form and Order

When a user tries to buy a product either directly from the product's page or the cart, the user will get a form to fill in all the data required in online shopping services like name, address, zip code, etc. The data obtained from this page will be stored and then the page will proceed further into the Stripe payment gateway, which was the used payment gateway option to implement the buying process.

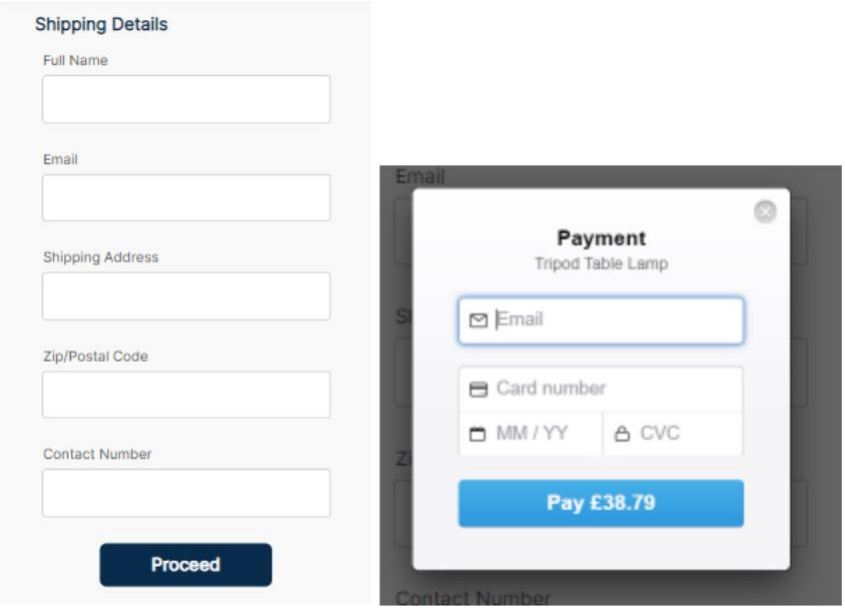


Figure 12: Booking form and payment popup

Stripe payment gateway will take the data from the chosen product and price of the item and it will be displayed in the payment popup and below code was used to pass the data to the gateway.

```
$('#customButton').on('click', function(e) {
  handler.open({
    name: 'Payment',
    currency: 'gbp',
    description: $('#item_name').val(),
    amount: $('#total_value').val() * 100
  });
  e.preventDefault();
});
```

Figure 13: Stripe payment implementation

After the submission of the card details user will receive relevant feedback according to the given card details.

2.7 Favourites list

Users can add products of their choice to their list of favourites and they can view their favourites list directly from their profile. On the product page, users can click on the heart icon and it will add the relevant product to the favourites list and it will be indicated by a filled heart icon and a toast message.

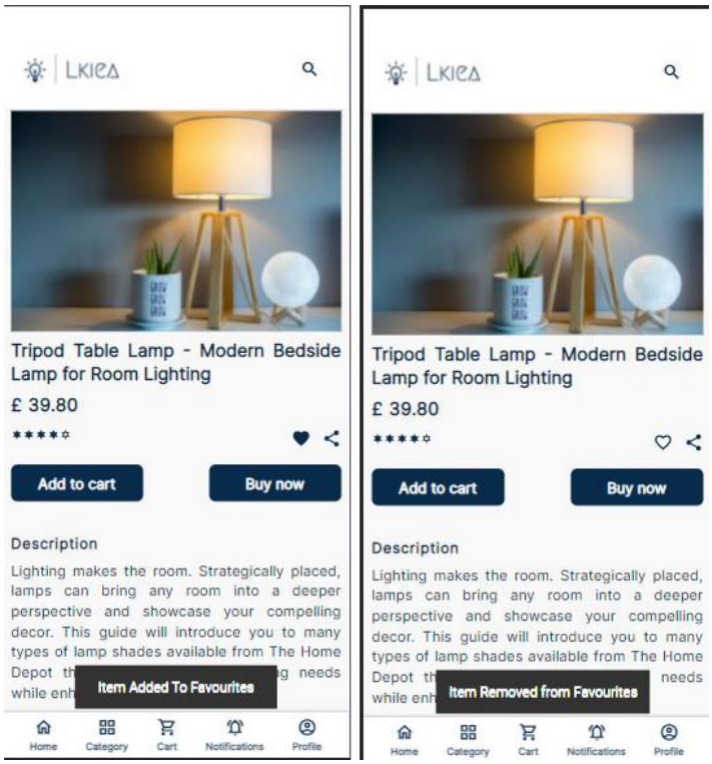


Figure 14: Favourites

When the user clicks on the product's heart icon all the required data from the product is gathered and sent to the Firebase platform and when the user wants to view all the favourited products then all the data for the relevant products will be fetched from the DB and shown as card views. Furthermore, users have the ability to remove a product from the favourite list as well. Below code was used for the interactions of data between frontend and backend.

```
let sendThis="";
productsRef.on("child_added", function(snap) {
  $('#display_fav').append(productHtmlFromObject(snap.val(),snap.key));
  $('#display_cart').append(productCartHtmlFromObject(snap.val(),snap.key));
  arr.push(snap.val());
  console.log("arrarr",arr);
});
```

```
function productHtmlFromObject(product,key){

  console.log(JSON.stringify(product)+"prod",key);
  remove_key=key;
  var str = product.main_title;
  if(str.length > 10) str = str.substring(0,19);
    var prd = '';
    prd += '<div id="id " class="prod-grid1">';
    prd += ' <a href="#new_product_page">';
    prd += ' <div class="card_elements" id="redirect-product">';
    prd += ' <div class="card-image">';
    prd +=
      ' <img alt="home"src="'+product.main_img+'"class="image_card"/>';
    prd += " </div>";
    prd += ' <h5 class="card-product-title">  ' + str;
    prd += ' <h6 class="card-product-desc">  ' + product.price;
    prd += " </h6>";
    prd += '<a href="#" class="card-product-desc-remove" data-key="'+key+'style="position: relative; left: 10px;" >Remove From Favourite</a>'
    prd += " </div>";
    prd += " </a>";
    prd += "</div>";
    return prd;
  }
```

Figure 15: Implementation of Favourites

From the favourites page, the user can share the list of favourite products via email to a valid email of the user's choice. When the user tries to send the email, using a regex email gets validated and if the email is invalid then the user will get a warning message when the email is valid then favourite product list fetched from the DB will be appended as an email body and will be shared using SmtpJS.

```
function sendEmail(mail) {
  sendThis+="\n === FAVOURITE LIST === \n";
  for (let i = 0; i < arr.length; i++) {
    sendThis+="\nItem Title - "+arr[i].main_title+"\n";
    sendThis+="Item Price - "+arr[i].price+"\n";
  }

  Email.send({
  Host: "smtp.gmail.com",
  Username : "lkialk2022@gmail.com",
  Password : "This@1234",
  To : mail,
  From : "lkialk2022@gmail.com",
  Subject : "Favourite List",
  Body : sendThis,
  Attachments : [
    {
      name : "lkia.png",
      path:"https://res.cloudinary.com/dgly8b9lq/image/upload/v1640846394/Lkia/main_p0wjoy.png"
    }]
  }).then(
    console.log()
  );
}
```

Figure 16: Implementation of email share

Favourite products can be viewed as card elements and when the orientation changes, to avoid empty white spaces cards will be rearranged.

2.8 Gamification (might need to be combined with 2.9)

A gamification process has been implemented in a separate section of the mobile web app to gather more customer attraction towards the use of the Lkiea application. In the application, we introduce a point system called Lkiea points which can be gathered by playing games that have been implemented in the Play & Win section.

Games were designed and implemented with the use of JavaScript and HTML and several functions and scripts were implemented to handle all the actions related to the game.

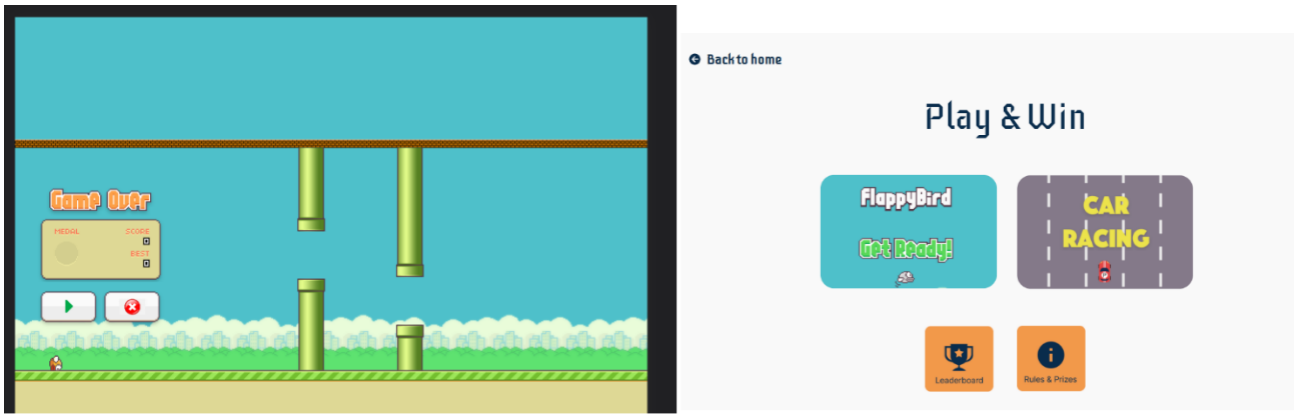


Figure 17: Gamification Process

2.9 System feedback

Few system feedbacks options have been implemented in the application in order to make the application more interactive and user-friendly. In most of the scenarios where the user has to interact with the system, popup messages, popup boxes, dialog boxes or toast messages were implemented. As an example, when the user adds and deletes a product from the favourite list a toast message will be displayed to the user informing about the change he/she made.

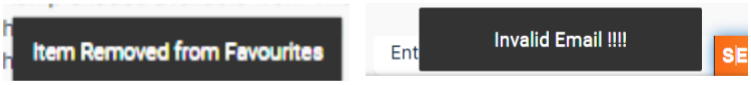


Figure 18: System feedbacks

As the above diagrams show simple English phrases were used in the feedback messages to avoid any confusion. Application has a white color-based theme hence all the feedback toast messages were created with a black background to increase the visibility and in the popup boxes also background overlay turns dark when the popup box appears on the screen. Logical aspects of the system feedback were handled mainly using JavaScript. Below codes were used when implementing the feedback messages.

```
<div data-role="popup" id="popupLogin" data-theme="a" class="ui-corner-all">
  <form method="post" id="reviewForm">
    <div style="padding: 10px 20px">
      <h3>Add your review and rating</h3>
      <input type="text" class="form-control" id="name" required placeholder="Enter name" />
      <textarea name="textarea-enhanced" id="email" data-enhanced="true"
        class="ui-input-text ui-shadow-inset ui-body-inherit ui-corner-all" required
        placeholder="Your Review"></textarea>
```

Figure 19: Feedback implementation

2.10 Scores & points gained (might need to be combined with 2.7)

The scoring system is directly linked to the gamification component of the implemented system. As mentioned earlier, the application introduces Lkiea point system that can be gathered after playing the game. Users can get a score for the game according to their performance and Lkiea points will be awarded according to their score.

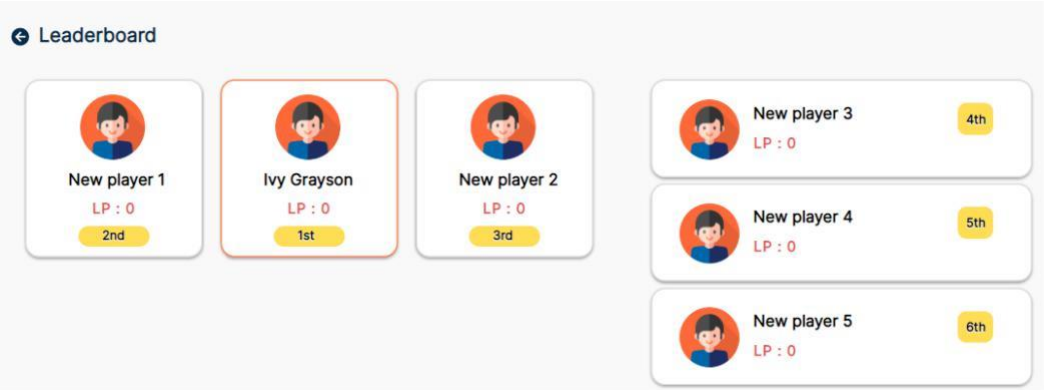


Figure 20: Leaderboard

The user can view all the other users who have played the game and scored the most by using the leaderboard option. The scoring system was implemented in a way so that, when the user exits after finishing the game the amount, he/she scored will be directly sent to the DB and updated on the leaderboard in real-time. Therefore, when the leaderboard is viewed all the data from the DB related to the leaderboard is fetched and shown in an interactive format.

```
scoreRef.on("child_added", function (snap) {
    $('#userScore').append(scoreHtmlFromObject(snap.val()));
    console.log("added", snap.val());
    ps = snap.val()
    scoreArray.push(ps.Points)
    console.log('snap val '+ps)

    latestScore = scoreArray[scoreArray.length - 1]
    // localStorage.setItem('latest score',latestScore)

    // console.log('latest score '+ latestScore)

});

function scoreHtmlFromObject(ps) {
    console.log(JSON.stringify(ps) + "scr");

    var scr = '';

    scr += 'LP : ' + ps
    console.log('ps ',ps)

    return scr;
}
```

Figure 21: Implementation of leaderboard

2.11 Comments

Comment option was implemented in the system with relation to the POIs. Users can visit a product POI and can add a comment in the review option or the ask question option. If the user can answer any question in the questions windows then the user can reply to the comment in real-time. Comments can be added using the text input fields

from the popup window. By clicking on the view more button in both reviews and questions user will be able to see all the reviews and questions posted about the relevant product with the posted date and time and can add a review, reply in real-time.

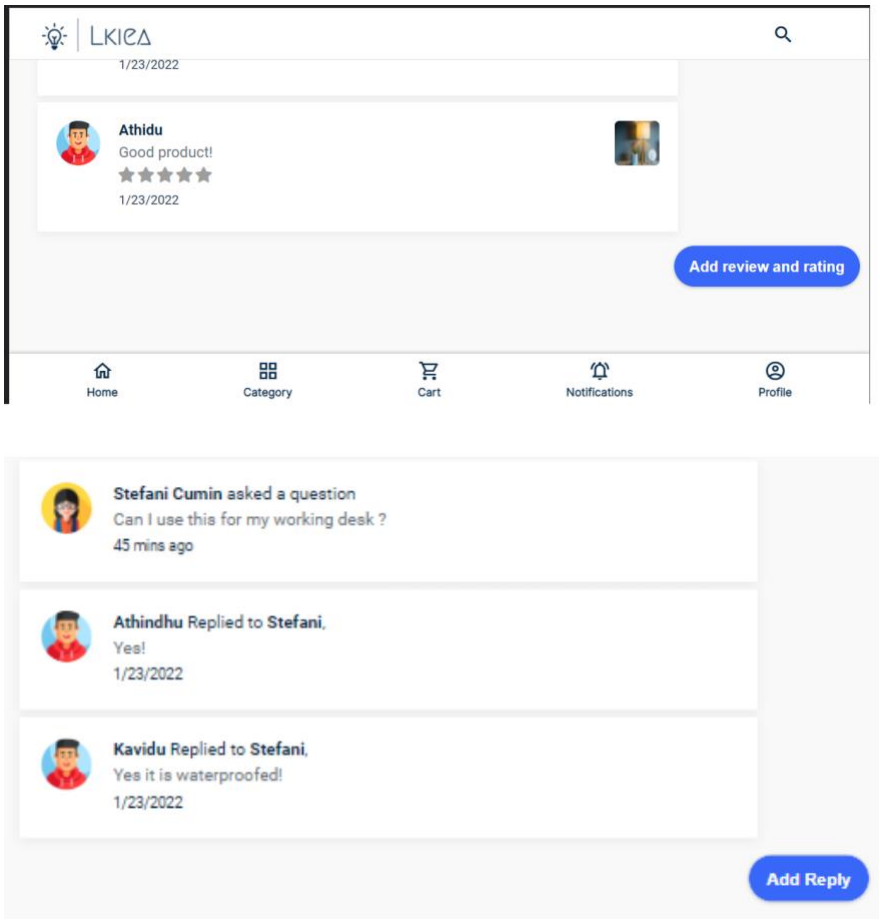


Figure 22: Comments, Reply

The comments section was built responsively to match all the device orientations, and the same font used throughout the application was used while a sample avatar image was used to show the user's profile.

When a user submits the form in the popup box the data will be retrieved as Strings and shared with Firebase real-time database and the added comment/reply/review will be added to the application in real-time. The below code is responsible for adding the comments.

```
//load older conatcts as well as any newly added one...
answersRef.on("child_added", function(snap) {
  console.log("added", snap.key, snap.val());
  $('#answers').append(answerHtmlFromObject(snap.val()));
});

//save contact
$('.addReply').on("click", function( event ) {
  //  $('#popupLogin').addClass('hidden');
    event.preventDefault();
    if( $('#name_ans_user').val() != '' || $('#user_answer').val() != '' ){
      answersRef.push({
        name: $('#name_ans_user').val().replace(/<[^>]*>/ig, ""),
        email: $('#user_answer').val().replace(/<[^>]*>/ig, ""),
        date: today.toLocaleDateString("en-US", options)
      })
      $('#answersForm')[0].reset();
    } else {
      alert('Please fill atlease name nn!');
    }
  });
```

Figure 23: Implementation of comments, reply

2.12 Rating

All the products listed in the application can be rated and a common star rating system was implemented in the Lkiea mobile web application. Users can add a rating to a product by using the view more option in the review & rating component under the Product page. Users can choose the number of stars they think the product should be rated from five stars and they can add the star value and submit.

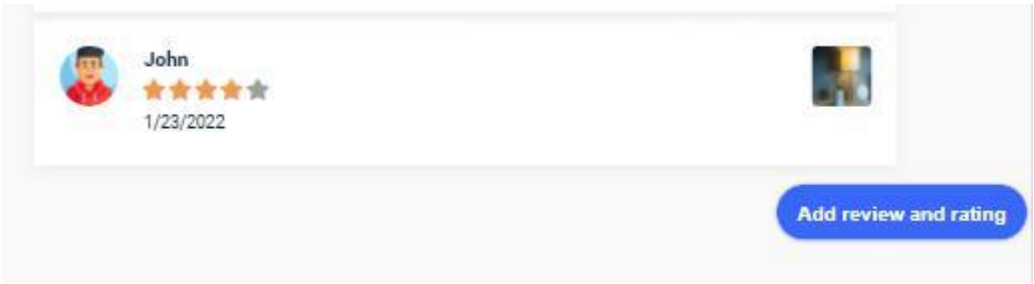


Figure 24: Rating

In this functionality also, the count of stars the user selected will be retrieved and shared with Firebase in real-time DB and star values will be retrieved and shown as a separate entity in the rating windows. In the rating windows, users can see all the ratings submitted by other users towards the same product as well. At the initial stage of the rating, all the stars will be empty in gray color and when the user hovers over or select the stars, then the selected number of stars will be in gold color.

Values related to the stars will be fetched from DB and will be shown as stars in the application view as shown below.

```
$('.stars a').on('click', function(){
  $('.stars span, .stars a').removeClass('active');
  let st = $(this).attr("id");
  console.log("stars+   " + st)
switch (st) {
  case "st-1":
    stars = 1;
    break;
  case "st-2":
    stars = 2;
    break;
  case "st-3":
    stars = 3;
    break;
  case "st-4":
    stars = 4;
    break;
  case "st-5":
    stars = 5;
    break;
  default:
    stars = 0;
    break;
}
  $(this).addClass('active');
  $('.stars span').addClass('active');
  console.log("stars+" + stars)
});
```

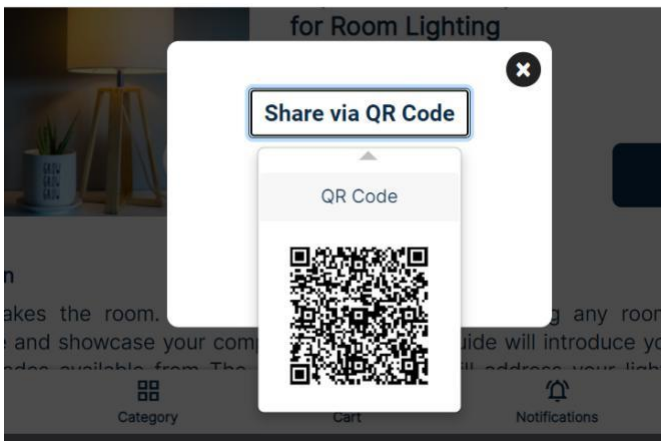
```
//prepare conatct object's HTML
function contactHtmlFromObject(contact){
console.log(JSON.stringify(contact)+"Jaye");
  let count = contact.star;

  var star = '';

  for (let i = 0; i < 5; i++) {
    if(count != 'undefined'){
      if(count > 0){
        star +=  '<span class="active">';
    star +=  '<a id="readonly_rating">1</a>';
    star +='</span>';
    count--;
      } else{
        star +=  '<span class="read-only">';
        star +=  '<a id="readonly_rating">1</a>';
        star +='</span>';
      }
  }}
```

Figure 25: Implementation of rating

2.13 QR / AR

QR, AR and VR technologies were implemented in the Lkiea mobile web applications to carry out the different tasks and to increase user attraction.

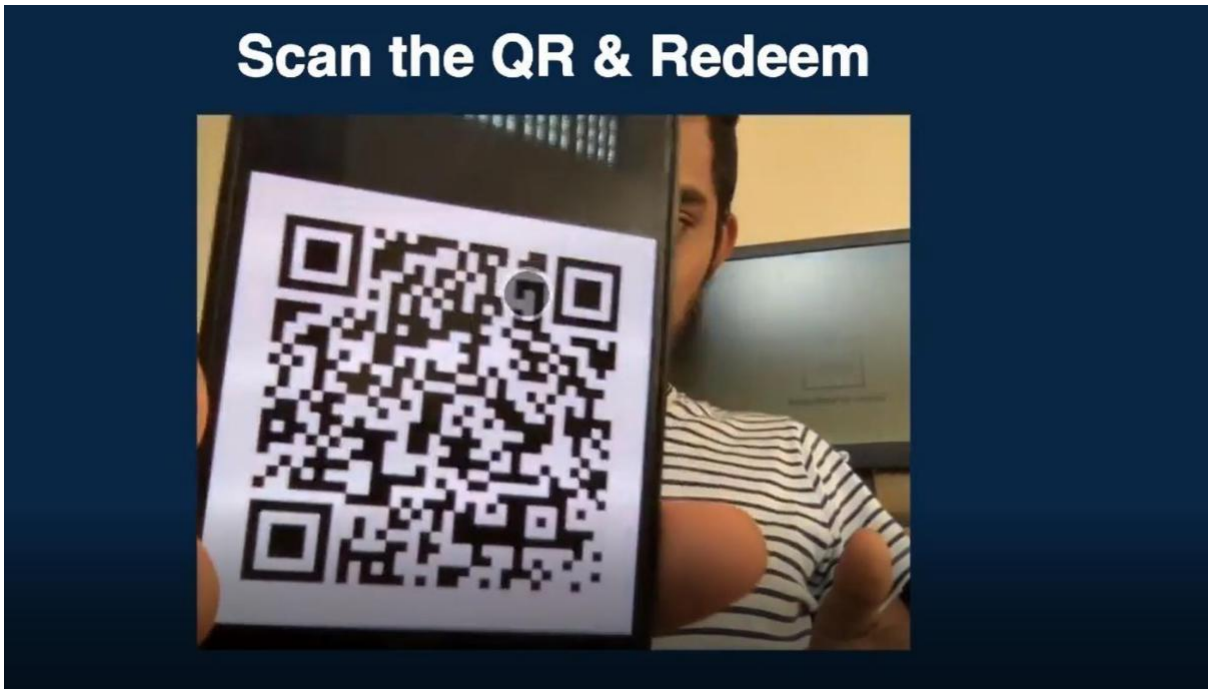
In the implemented system users can both scan QR codes and generate QR codes. If collaborated further, from the Product page users can share the specific product as a QR code, and by sharing this generated QR code anyone can access the specific product after scanning it and for generating the QR code, a JavaScript QR code library was used.



```
<script>
  var qrcode = new QRCode(document.getElementById("qrcode-view-pop"), {
    width: 120,
    height: 120
  });
  function makeQrcode(e) {
    qrcode.makeCode(window.location.href);
  }
  jQuery(document).ready(function () {
    jQuery("[data-toggle='popover']").popover(
      options = {
        content: jQuery("#qrcode-view-pop"),
        html: true // important! popover html content (tag: "#qrcode") which contains an image
      }
    );

    jQuery("[data-toggle='popover']").on("show.bs.popover", function (e) {
      makeQrcode(jQuery(this));
      jQuery("#qrcode-view-pop").show();
    });
  });
</script>
```

Figure 26: QR Code Generate

Furthermore, users can scan QR codes to redeem coupon codes that are implemented on the profile page. Users can show the QR code to the camera and get the output in the same window and by using the copy option implemented there, users can easily copy and apply the output. QR reader implementation was carried out with the use of an inbuilt JavaScript QR code library.



```
qrcodeW.callback = (res) => {
  if (res) {
    outputData.innerText = res;
    scanning = false;

    video.srcObject.getTracks().forEach(track => {
      track.stop();
    });

    qrResult.hidden = false;
    btnScanQR.hidden = false;
    canvasElement.hidden = true;
  }
};

function tick() {
  canvasElement.height = video.videoHeight;
  canvasElement.width = video.videoWidth;
  canvas.drawImage(video, 0, 0, canvasElement.width, canvasElement.height);

  scanning && requestAnimationFrame(tick);
}

function scan() {
  try {
    qrcodeW.decode();
  } catch (e) {
    setTimeout(scan, 300);
  }
}
```

Figure 27: QR Reader

The augmented reality component also includes a VR component and an AR library named aframe.io was used in the implementation of this component. From the popup window in the profile scan and redeem users can select the AR/VR component and it will redirect the user to a page implemented with VR. VR components were developed using 3D models and a-frame and users can traverse through the space and experience VR implementation. In VR component 2 easily visible 3d mystery boxes and one hidden mystery crate has been implemented and by clicking on each one of these boxes users will be redirected to a page with AR and by showing Hiro marker to the AR camera user will be able to see coupon codes, promotional messages, etc.

The below code was used when implementing the VR/AR feature.

```
<a-scene cursor="rayOrigin: mouse" >
  <a-assets>
    <a-asset-item id="mBox" src="../assets/mystreyBox.glb"></a-asset-item>
    <a-asset-item id="selectTextBox" src="../assets/select.glb"></a-asset-item>
    <a-asset-item id="mysteryCrate" src="../assets/crate.glb"></a-asset-item>
    <img id="panoramaQR" src="../assets/stock.jpg" />
  </a-assets>

  <!-- Text title -->
  <a-entity gltf-model="#selectTextBox" scale="20 20 8" position="4 5 2">
  </a-entity>

  <!-- Box code 1 -->
  <a-entity
    gltf-model="#mBox"
    scale="4 4 4"
    position="0 1 2"
    id="mBox-code1"
    mylink="href: ARCode1.html;"
  >
    <a-animation
      attribute="rotation"
      to="0 360 0"
      repeat="indefinite"
    >
    </a-animation>
  </a-entity>
```

```
<a-scene embedded arjs='sourceType: webcam;'>
    <a-entity
          geometry="primitive: plane; width: 5; height: auto" rotation="-90 0 0"
          material="color: #092C4C"
          text="value: CONGRATS !!! -----------YOU won 10€ coupon-----------  Your Coupen Code is: 735 892 ">
    </a-entity>

    <a-marker-camera preset='hiro'></a-marker-camera>
</a-scene>
```

Figure 28: AR/VR

2.14 Differentiation between iPhone and iPad implementation

User interfaces have been designed to avoid white space and follow the design principles, and due to the huge size difference between the two devices and two orientations some pages of the mobile web application were designed specifically to match the device and its orientation.

And between iPhone and iPad popup boxes were separately resized according to the screen size of the device. Apart from these few other differentiations can be observed from the design aspect of the two implementations.

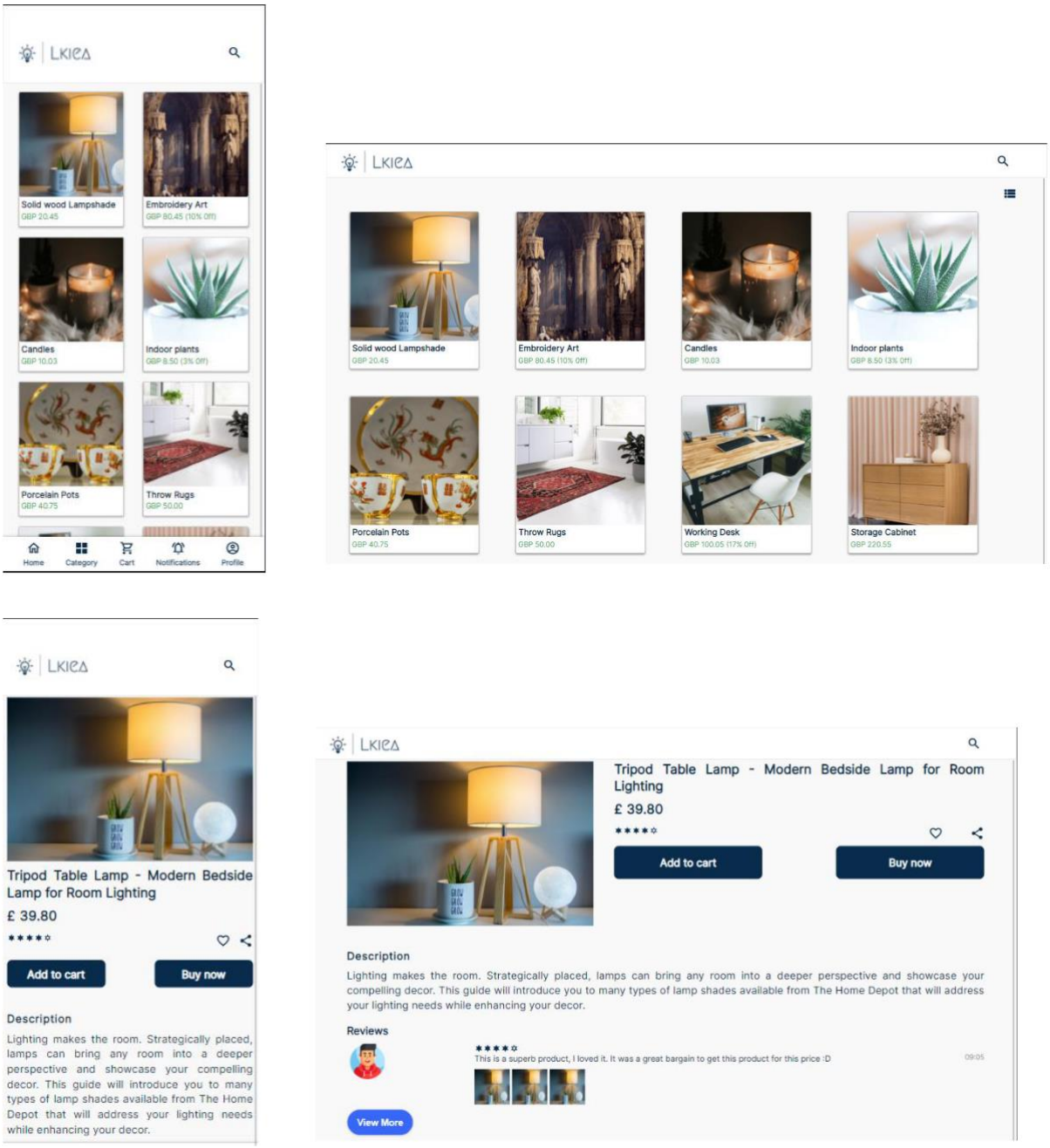


Figure 29: iPhone vs iPad Comparison

References

smtp.js - https://smtpjs.com/

jQuery Mobile - https://demos.jquerymobile.com/1.4.5/

Firebase - https://firebase.google.com/docs