Senni Ojaniemi

FLUTTERIN JA REACT EXPON VERTAILU

Alustariippumattoman mobiilisovelluskehitysalustan valinta yrityksessä

Opinnäytetyö
CENTRIA-AMMATTIKORKEAKOULU
Tieto- ja viestintätekniikan koulutusohjelma
Joulukuu 2020

TIIVISTELMÄ OPINNÄYTETYÖSTÄ

Centria-ammattikorkeakoulu	**Aika** Joulukuu 2020	**Tekijä/tekijät** Senni Ojaniemi
Koulutusohjelma Tieto- ja viestintätekniikka		
Työn nimi FLUTTERIN JA REACT EXPON VERTAILU. Alustariippumattoman mobiilisovelluskehitysalustan valinta yrityksessä		
Työn ohjaaja Jari Isohanni		**Sivumäärä** 24 + 5
Työelämäohjaaja Jari Isohanni		

React Nativen aloittelijatyökalu React Expo ja Flutter ovat molemmat suosittuja sovelluskehitysalustoja alustariippumattomille mobiilisovelluksille. Mobiilisovelluskehitystä aloittelevassa yrityksessä niiden välillä valinta voi olla vaikeaa, ja internetin tarjoamissa lähteissä vastaus jätetään usein avoimeksi sekä tilannekohtaiseksi. Työssä selvitettiin, kumpi kehitysalusta sopisi yritykselle paremmin valinnan ollessa epäselvä, ja pitävätkö medialähteissä tehdyt huomiot paikkansa valintaa tehdessä.

Työssä tehtiin React Expolla ja Flutterilla yksi sovellus, eli yhteensä kaksi sovellusta saman suunnitelman pohjalta. Medialähteiden huomiot havaittiin suurimmalta osin paikkansa pitäviksi. Sovelluskehityskokemusten jälkeen Flutter koettiin paremmaksi vaihtoehdoksi alustariippumatonta sovelluskehitystä aloittelevalle yritykselle, koska sen yksinkertainen asentaminen, ongelmanratkaisumenetelmien selkeys ja nopea tulosten visuaalinen näkyvyys olivat parempi valinta aloittelijalle. React Expo koettiin hyväksi valinnaksi vain, jos projektin tarpeet ja resurssit olivat selkeästi enemmän kyseisen sovelluskehitysalustan puolella.

Asiasanat
Alustariippumaton sovelluskehitys, Flutter, mobiilisovelluskehitys, React Expo, React Native.

ABSTRACT

Centria University of Applied Sciences	Date December 2020	Author Senni Ojaniemi
Degree programme Information Technology		
Name of thesis COMPARING FLUTTER AND REACT EXPO. Choosing a cross-platform development platform in a corporation		
Instructor Jari Isohanni		**Pages** 24 + 5
Supervisor Jari Isohanni		

The Beginner tool of React Native; React Expo and Flutter are both popular development platforms for cross-platform mobile apps. In a corporation still beginning their mobile application development, choosing between the two can be difficult, and the sources provided through the Internet often leave their conclusions open and dependent on context. This work examines which development platform would be better suited for a company when the choice is unclear, and whether the observations made in the media can be depended on when making the decision.

In this work, one application was made with React Expo and Flutter, a total of two applications based on the same plan. The findings from media sources were found to be largely correct. After software development experience, Flutter was perceived as a better option for a beginner cross-platform company because of its simple installation, clarity in problem-solving and fast acquisition of visual results. React Expo was considered a good choice only if the needs and resources of a company lean clearly towards it.

Key words
Cross-platform software development, mobile application development, Flutter, React Expo, React Native.

KÄSITTEIDEN MÄÄRITTELY

Alustariippumaton sovellus
Sovellus, joka toimii useassa eri käyttöjärjestelmässä vain yhdellä lähdekoodilla.

MVP-sovellus
(Minimum Viable Product) on sovelluskehitysmenetelmä, jossa tehdään minimaalisesti toimiva sovellus, jossa on vain ydintoiminnot. Sovelluksen tarkoituksena on ratkaista tietty ongelma kehityksessä tai luoda tyydyttävä käyttökelpoinen sovellusversio asiakkaalle esiteltäväksi.

Natiivi sovellus
Sovellus, joka toimii samalla lähdekoodilla käyttölaitteen käyttöjärjestelmän kanssa.

SDK-paketti
(Software development kit.) Lajitelma sovelluskehitystyökaluja, jotka asennetaan yhdessä paketissa.

Sovelluskehitysalusta
Teknologiakokonaisuus, joka auttaa kehittäjää suunnittelemaan, kehittämään ja toteuttamaan sovelluksiaan. Esimerkiksi Visual Studio.

UI-työkalupaketti
(User Interface -työkalupaketti) Työkalupaketti, jossa on joukko rutiineja ja apuohjelmia, jotka tarjoavat kehittäjälle työkaluja käyttöliittymän luomiseen sovelluksessa. Työkalupaketti luo ja ylläpitää syötettä, kehotteita, viestejä, valikoita, ohjeita, ja muita käyttöliittymän kohtia, joita kehittäjän ei tarvitse kirjoittaa koodiin itse.

UML-kaavio
(Unified Modeling Language) on tapa visualisoida sovellus erilaisten diagrammien avulla.

1 JOHDANTO

Mobiilisovellukset ovat nykyään suosituin tapa, jolla käyttäjät yhdistävät internetiin. Monet yritykset etsivät tapoja pysyä käyttäjien huomiossa omalla mobiilisovelluksella, joka tavoittaa mahdollisimman monta käyttäjää. Hyvä tapa kehittää nopeasti mobiilisovellus mahdollisimman laajan käyttäjäyhteisön saataville on alustariippumattomat sovelluskehitysalustat mobiilisovelluksille. (IBM Cloud Education, 2018.) Mikä alusta yrityksen tulisi valita monien eri vaihtoehtojen joukosta, ja millä perusteella?

Kaksi suosituinta isojen yritysten ylläpitämää mobiilisovelluskehitysalustaa ovat React Native ja Flutter (Galadzhii, 2020.). Niiden välillä on paljon vertailuita saatavilla, mutta harvoissa sanotaan suoraan, kumpi on parempi valinta yritykselle, joka haluaa kehittää ensimmäisen mobiilisovelluksensa. Kehitysalustoilla on paljon samankaltaisuuksia, ja hyviä että huonoja puolia, joten yrityksen, jolla ei ole mobiilisovelluskehitystaustaa, voi olla vaikea tehdä valintaa noin vain. Kumpi siis on parempi, jos minkäänlaista taustatietoa ei ole? Kumman valinta olisi yritykselle tuottoisampi ja helpompi, ja millaisissa tilanteissa toinen kehitysalusta on parempi valinta kuin toinen?

Työssä ohjelmoitiin kummallakin kehitysalustalla saman suunnitelman pohjalta yksi mobiilisovellus. Ohjelmoitaessa vertailtiin molempien ympäristöjen käyttökokemusta ja soveltuvuutta ensimmäisen mobiilisovelluksen kehitykseen. Työssä tutkitaan, kumman käytön oppiminen ja jatkaminen olisi kannattavampaa yrityksessä, joka ei ole varma kehitysalustan valinnasta. Päätavoitteena on selvittää, kumman kehitysalustan valitseminen olisi kannattavampaa ja pitävätkö saatavilla olevat nettivertailut paikkansa.

2 MOBIILISOVELLUSTEN KEHITYS

Mobiilisovellus on mobiililaitteelle, eli yleisimmin älypuhelimelle tai tabletille, suunniteltu sovellus (Kielitoimiston sanakirja, 2020). Mobiilisovellusten kehityksellä tarkoitetaan siis sellaisten ohjelmistojen kehittämistä, suunnittelua ja toteuttamista, joita tullaan käyttämään pääosin mobiililaitteilla. Sovellusten kehityksessä pyritään käyttämään sellaisia ohjelmointikieliä, työkaluja ja alustoja, joilla voidaan tarjota mahdollisimman nopea, helposti muokattava, pieniresurssinen ja helppokäyttöinen sovellus. (IBM Corporation, 2011.)

Mobiilisovelluksia kehitetään erilaisilla alustoilla ja ohjelmointikielillä sovelluksen käyttötarkoituksen mukaan. Yleisimpiä ohjelmointikieliä mobiilisovellusten kehityksessä ovat muun muassa Java, Swift ja Javascript. (Bura, Cajthaml, Hasa, Kraft & Vavra, 2020.) Mobiilisovelluskehitykseen on tarjolla monia sovelluskehitysalustoja eri tarkoituksiin, esimerkiksi Xamarin ja Appy.io (Gupta, 2019.) Työssä käytetään sovelluskehitysalustana mobiilisovellusten kehitykseen Android Studiota ja Visual Studiota.

Mobiilisovellukset muuttuvat jatkuvasti ajan myötä. Niiden käyttämää koodia lisätään, poistetaan ja muokataan jatkuvalla syötöllä, minkä takia järjestelmät muuttuvat tasaisesti. Mobiilisovelluksen elinkaari poikkeaa esimerkiksi työpöytäsovellusten elinkaaresta monimutkaisuuden näkökulmasta. (Zhang, Sagar & Shihab, 2013, 4–7.) Tämän takia yritysten voi olla vaikea tietää, mikä mobiilisovelluskehitysalusta kannattaa ottaa käyttöön. Työssä on valittu vertailtaviksi Flutter ja React Nativen käyttöönottotyökalun React Expo niiden markkinasuosion perusteella (Galadzhii, 2020).

2.1 Flutter

Flutter on Googlen UI-työkalupaketti, jolla voi kehittää sovelluksia mobiililaitteille, verkkoon ja tietokoneelle yhdellä koodipohjalla. Flutterin toiminta perustuu nopeaan sovelluskehitykseen, ilmaisevaan ja joustavaan käyttöliittymään sekä laitteen sisäisen suorituskyvyn hyödyntämiseen. Flutterin UI-työkalut eli widgetit ovat täysin muokattavissa, ja ne ottavat huomioon alustakohtaiset suorituskykyrajoitukset. Flutter-sovelluksissa saattaa esimerkiksi olla erilaiset vieritysanimaatiot käyttöalustan mukaan. Flutterin toiminta perustuu Dart-koodin käyttöön, joka kootaan käyttölaitteen natiiviksi koodiksi. Tämän takia Flutter-sovellukset toimivat tuetuilla käyttöalustoilla käyttäen vain yhtä koodia. (Flutter, 2020a.)

Flutter on avoimen lähdekoodin projekti, jota edistävät Google ja Flutterin kehitysyhteisö. Se on tarkoitettu olio-ohjelmointiin suuntautuneille ohjelmoijille helposti lähestyttäväksi työkaluksi mobiilisovellusten koodaukseen. Flutter on optimoitu 2D-teknologiaa hyödyntäville interaktiivisille sovelluksille, joten 3D-sovelluksien tuottamiseen se ei esimerkiksi sovellu. Flutter-sovelluksen runko on kerroksittainen ja muokattavissa, ja se käyttää minimaalisen määrän C- ja C++-koodia. Muuten Flutter käyttää täysin omaa ohjelmointikieltään Dartia. (Muller, 2020.)

Flutterilla kehitettyjä sovelluksia käyttävät esimerkiksi Google, eBay ja BMW (Flutter, 2020b). Flutterin tiimi uskoo, että sovelluskehittäjien on suunnattava palveluitaan mahdollisimman monelle eri mobiilialustalle, mutta HTML ja WebView tekevät korkealaatuisen sovelluksen kehittämisestä haastavaa. Nykyään on liian kallista ohjelmoida sama sovellus monta kertaa, jotta sen saa uusien alustojen markkinoille. Flutter tarjoaa parempaa tapaa kontrolloida yhtä koodipohjaa useille eri alustoille uhraamatta suorituskykyä ja sovelluksen laatua. Flutter keskittyy kontrollin antamiseen sovelluskehittäjille, sovellusten hyvään suorituskykyyn ja luotettavaan käyttökokemukseen sekä ohjelmoijien että käyttäjien näkökulmasta. (Muller, 2020.)

2.2 React Expo

React Expo on React Nativen rakennetyökalusarja, jolla voi luoda React Native alustariippumattomia sovelluksia. Se on suunniteltu mobiilisovelluskehityksen helppoon ja nopeaan aloittamiseen. Siitä kuitenkin puuttuvat mukautetut, natiivit moduulit, jotka React Native CLI, React Nativen kokonainen mobiilisovelluskehitys työkalusarja, tarjoaa. (Facebook Open Source, 2020.)

React Expo on rakennetyökalusarja ja alusta laaja-alaisille React Native -sovelluksille. Sen työkalut ja palvelut on rakennettu React Nativen ja natiivien alustojen ympärille, ja se toimii JavaScript- tai Typescript-koodilla. (Expo Team, 2020.) React Expo on ilmainen, avoimen lähdekoodin työkalusarja (Expo Team, 2020), jota Facebook tukee (Facebook Open Source, 2020). React Expo hyödyntää käyttölaitteen natiiveja toimintoja ja tarjoaa yhden koodipohjan usealle eri alustalle. React Nativella on laaja kommuuni ja monia kehittäjän työkaluja. Sovelluksen julkaisun jälkeen päivitykset tulevat käyttäjien saataville reaaliajassa. (Expo Team, 2020.)

React Native -sovelluksia ovat esimerkiksi Facebook, Instagram ja Discord. React-sovelluskehitys perustuu vahvasti yhteisön kehitykseen, ja sen useat kommunikaatioalustat, kuten kekustelupalstat ja

Slack-ryhmä, ovat todella aktiivisia. (Facebook Open Source, 2020.) Koska työssä tehtävät testisovellukset ovat yksinkertaisia ja työ on pääosin aloittelijan näkökulmasta, vertailukohteena käytetään React Expoa esimerkiksi React Native CLI:n sijaan.

3 FLUTTERIN JA REACT EXPON VERTAILU VERKOSSA

Monet yritykset ja yksityiset sovelluskehittäjät etsivät työkaluja alustariippumattomaan mobiilisovelluskehitykseen. Koska Flutter ja React-perhe ovat molemmat isojen yrityksien tukemia, ne ovat suosituimpia vaihtoehtoja. (Galadzhii, 2020.) Internetin tarjoamaa tietoa tutkimalla pyritään ottamaan selvää, tullaanko työn sovelluskehitysvaiheen jälkeen samoihin lopputuloksiin.

Suoraa vertailua React Expon ja Flutterin kanssa ei ole, vaan vertailijat viittaavat suoraan yleisesti React Nativeen ja kaikkiin sen kehitysalustan alamuotoihin, myös Expoon. Täten React-perheen sovelluskehitystyökalut tulevat vertailtaviksi kokonaisuudessaan, eivätkä vain aloittelijatyökalun näkökulmasta. Verkkovertailussa on otettava huomioon, että React Nativea vertaillaan laajemmasta ominaisuusnäkökulmasta kuin työssä.

Ravichandran (2019) jakaa Flutterin ja React Nativen valitsemisen eri tekijöiden välille. Valintaperusteina toimivat ohjelmointikieli, yritystuki, trendit, suorituskyky, käyttöönottojen määrä ja työllistäminen. Ohjelmointikielen voittaa React Native JavaScript-kielellä, joka on yksi suosituimmista ohjelmointikielistä ympäri maailman. Flutterin käyttämä Dart on helppo oppia C++- tai Java-taustalta, mutta vain Flutter käyttää kyseistä ohjelmointikieltä, joten JavaScriptin opettelu on hyödyllisempää. Molempia ohjelmointikieliä tukevat suuret yritykset, mutta vanhempana sovelluskehitysalustana React Nativea käytetään yrityksissä enemmän, joten se tarjoaa siten enemmän työpaikkoja markkinoilla. Vuoden 2019 trendien perusteella Flutter on React Nativea suositumpi, ja se on kerännyt suosiota Reactiin verrattuna lyhyessä ajassa. Myös Flutterin suorituskyky on parempi sen käyttämän natiivin koodin ansiosta. Ravichandran on itse työskennellyt React Native sovellusten parissa, mutta voittajan sijaan päättää valinnan olevan tapauskohtainen ohjelmoijan edellisen ohjelmointitaustan perusteella. (Ravichandran, 2019.)

Mroczkowska, Skuza ja Włodarczyk (2019) tutkivat sovelluksen omistajan näkökulmasta, kumpi sovelluskehitysalusta kannattaisi valita mieluummin vuonna 2020. Flutterin ja React Nativen vertailu jaetaan plussiin ja miinuksiin. Flutterin hyviä puolia heidän mukaansa ovat Hot Reload -toiminto, yhden koodipohjan täysi hyödyntäminen, testauskertojen vähäisyys, nopea suorituskyky, kauniit visuaaliset ominaisuudet, UI:n samankaltaisuus eri alustoilla ja helppous tehdä MVP-sovelluksia. Flutterin huonoiksi puoliksi listattiin sovelluskehittäjäyhteisön pieni koko, kirjastojen ja tuen vähäinen kirjo, jat-

kuvan integraation tuen puute, alustan riski loppua ja sovellusten suuri koko. Reactin hyviä puolia olivat Fast refresh -toiminto, yhden koodipohjan täysi hyödyntäminen, JavaScriptin käyttö, sovelluskehittäjän vapaus valita toimintoja ja asetuksia, suhteellisen hyvä kypsyysaste, aktiivinen ja iso kehittäjäyhteisö, helppo oppia verkkokehityspohjalta ja testauskertojen vähäisyys. Reactin huonoiksi puoliksi listattiin natiivin suorituskyvyn puute, vähäinen yleisten komponenttien määrä, sovelluskehittäjän vapaan valinnan vaikeus sekä sopivien työkalujen löytäminen, suuri hylättyjen sekä huonolaatuisten kirjastojen määrä, UI:n hauraus sekä sovellusten iso koko. Loppujen lopuksi oikean sovelluskehitystyökalun valinta riippuu kehitysryhmän taidoista. Jos ohjelmoijat tuntevat Dartin, Flutter on hyvä valinta. Jos ohjelmoijat tuntevat JavaScriptin, React Native on hyvä valinta. Brändin ensimmäisen sovelluksen muotoiluun Flutter on heidän mielestään paras valinta. (Mroczkowska, Skuza & Włodarczyk, 2019.)

Nader (2020) vertailee Flutteria ja React Nativea ensin kertomalla niiden ominaisuuksista ja sen jälkeen listaamalla plussat ja miinukset. Flutterin positiivisia puolia hänen mielestään ovat nopean koodauksen tukeminen, yksi koodipohja monella eri alustalla, vähemmän testaustarvetta, nopea sovelluskehitys, käyttäjäystävällinen UI-mallinnus, soveltuvuus MVP-sovelluksiin ja vähäinen koodin määrä. Negatiivisia puolia olivat sovelluskehittäjäyhteisön pieni koko, natiivin koodin tuottavuuden alittaminen, jatkuvan sovelluskehityksen tuen puute, sovelluskehitysalustan riskinalaisuus ja sovellusten koko. React Nativen positiiviset puolet olivat nopean sovelluskehityksen tukeminen, yksi koodipohja eri alustoille, JavaScriptin käyttö, suuri valinnanvapaus koodin uudelleenkäytössä, verrannollinen kypsyysaste, aktiivinen ja suuri kehittäjäyhteisö, helppo oppimiskäyrä, vähäinen testaustarve ja tukeva suorituskyky. Negatiivisia puolia olivat natiivisuuden puute, yleisten komponenttien vähäisyys, sovelluskehittäjän valinnanvaikeus, hylättyjen pakettien ja kirjastojen määrä, hauras UI ja sovellusten koko. Nader ei anna suoraa vastausta, kumpi on hänen mielestään parempi vaihtoehto, mutta sanoo Flutterin olevan mobiilisovelluskehityksen tulevaisuus ennustusten mukaan. Artikkelin päämäärä on auttaa yksilöä valitsemaan sopiva kehitysalusta omien tarpeiden mukaisesti. (Nader, 2020.)

Suurimmassa osissa lähteitä ei anneta suoraa vastausta, kumpi sovelluskehitysalusta on kirjoittajan mielestä parempi. Flutterin mainitaan soveltuvan hyvin MVP-sovelluksiin, mutta suoraa vastausta siihen, kumpaa sovelluskehitysalustaa esimerkiksi aloittelijan kannattaisi lähestyä, ei mainita. Artikkeleiden mukaan valinta riippuu projektin vaatimuksista, kehittäjätiimin taidoista ja yksilön omasta sovelluskehitystaustasta. Artikkeleiden mukaan työn aikana Flutterin pitäisi nousta paremmaksi vaihtoehdoksi C++- ja Java-ohjelmointikielien taustan takia.

4 SOVELLUSKEHITYS JA SEN EROT

Mobiililaitteet ovat käyttäjien suosituin tapa yhdistää internettiin. Yritykset tarvitsevat mobiilisovelluksia pysyäkseen asiakkaan näkökulmasta merkittävinä, vastaanottavina ja menestyvinä. (IBM Cloud Education, 2018.) Tutkimalla sovelluskehitysprosessia Flutterilla ja React Expolla pyritään huomioimaan, mitä eroja sovelluskehittäjän kannattaa ottaa huomioon ennen alustan valitsemista.

React Expo sekä Flutter täytyi molemmat asentaa komentokehotteen kautta tavalla tai toisella. Expon asentamiseen tarvitsi komentokehotteessa toimivan pakettienhallintatyökalun, jonka avulla Expo asennettiin (Expo Team, 2020). Flutterin viralliselta verkkosivulta sai ladattua ohjelman, joka avasi Flutterin hallintaikkunan komentokehotteeseen, josta Flutterin pystyi asentamaan ja asennuksen varmistamaan (Flutter, 2020a). Molempien kehitysympäristöjen verkkosivuilla oli selkeät ohjeet asentamisen helpottamiseksi, mutta molempien alustojen asentaminen vaati enemmän kuin aloittelijatason taitoja komentokehotteen ja pakettityökalujen käytöstä. Sovelluskehittäjälle, jolla ei ole kokemusta pakettien asentamisesta komentokehotteen kautta, molempien alustojen asentaminen voi olla vaikeaa.

Expon asennus sujui helposti ohjeiden mukaisesti NPM:n avulla, ja koska kehitystietokonella oli jo NPM-paketinhallintaohjelmisto asennettuna, Expon käyttöönotto onnistui nopeasti. Uuden Expo-projektin luominen onnistui NPM-komennoilla ohjeiden mukaisesti, ja testilaitteelle sai nopeasti eteen tyhjän pohjan, jolta aloittaa sovelluskehitys. (Expo Team, 2020.) Kehittäjä joutuu kuitenkin asentamaan Node.js-työkalut ennen Expon käytön aloittamista, ja aloittelevien sovelluskehittäjien pitää asentaa pakettienhallintatyökalu, NPM tai Yarn, ennen Expon käyttöönottoa. Täysin puhtaalta pöydältä aloittaessa Expo vaatii monia kolmansien osapuolien työkaluja ennen käytön aloittamista ja onnistunutta asennusta.

Flutterin asennus vaati Flutterin SDK-paketin lataamisen virallisilta sivuilta. Flutterin konsolin sai avattua ladattavan zip-tiedoston purkamisen jälkeen, ja Flutter asensi flutter doctor -komennon syötön jälkeen. Listatut ohjelmistot ja lisäosat pystyi päivittämään Android Studion avulla, ja käyttöehtojen hyväksymisen jälkeen Flutter oli käyttövalmis. Kehitystietokoneella oli jo asennettuna Android Studio ja Visual Studio Code, joten ne täytyi vain päivittää, ja lisätä Flutter-lisäosat sen käyttöönottoon. (Flutter, 2020a.) Alkuasennuksen jälkeen Flutterin ylläpidon pystyi hoitamaan Android Studion kautta, kun taas Expon päivittäminen täytyi hoitaa NPM:n avulla manuaalisesti. Expoon verrattuna Flutter vaati vähemmän kolmansien osapuolien ohjelmia ja niiden asennusta ennen käytön aloittamista.

Molempien työkalujen käyttö vaatii aiempaa koodauskokemustaustaa vähintään yhdestä ohjelmointikielestä, sillä molempien alustojen ohjeet oli suunnattu jo ohjelmoinnin pääperiaatteet tunteville henkilöille. Expon käyttämä JavaScript on yleisesti tunnettu ohjelmointikieli, mutta jos ei ole sen harjaantunut käyttäjä, oppiminen vie aikaa. Flutterin käyttämän Dartin opettelun pystyy aloittamaan kokonaan Flutterin tarjoamien tutoriaalien avulla. Dartin oppiminen onnistui helpommin kuin JavaScriptin, koska se on samankaltainen Javan ja C++ kanssa, ja kyseisistä ohjelmointikielistä on työtä ennen paljon kokemusta. JavaScript oli haastavampaa yhdistää aiemman osattuihin ohjelmointikieliin, joten sen hallitseminen vei enemmän aikaa. Sovelluskehityksen aikana sai kuitenkin hyvät käyttötaidot kumpaankin kieleen, mutta JavaScriptin opettelussa meni Dartiin verrattuna enemmän aikaa.

Sovelluskehityksen käyttökokemuksia vertailtaessa pyrittiin minimoimaan muut tekijät, jotka saattoivat vaikuttaa käyttömukavuuteen. Flutteria käytettiin Android Studiolla ja Expoa Visual Studio Codella, jotka ovat entuudestaan tuttuja sovelluskehitysalustoja. Molempia kehitysympäristöjä voi tarpeen mukaan ajaa, testata ja päivittää komentokehotekomentojen kautta ilman sovelluskehitysalustoja, mutta jos kehittäjä on tottunut sovelluskehitykseen sovelluskehitysalustoilla, on parempi käyttää niitä. Expo kuitenkin vaati NPM-komentokehoteikkunan testausta varten, kun taas Flutter vaati vain Android Studion testaukseen. Koodin ongelmien löytämiseen ja sovelluksen testaamiseen Expo vaati kolmansien osapuolien ohjelmia koodin muokkaamiseen ja virtuaalilaitteen ajamiseen, kun taas Flutterin käyttö onnistui saumattomasti Android Studiosta.

Molemmat kehitysympäristöt ovat jatkuvan kehityksen ja parantelun alaisena, joten molemmissa on vanhentuvia, muuttuvia sekä käytöstä poistuvia komentoja ja ominaisuuksia. Valmiita sovelluksia, koodipohjia ja pitkäaikaisia projekteja on aktiivisesti päivitettävä, jotta käytössä oleva koodi pysyisi ajan tasalla kehitysympäristöjen kanssa. Käytöstä poistettujen tai poistuvien ominaisuuksien käyttö ei kuitenkaan ole mahdotonta. Niiden käytön voi sallia uudelleen poikkeusmerkintöjenavulla, mutta niitä käyttäessä koodin ei enää taata olevan vakaa. (Flutter, 2020a; Expo Team, 2020.) Projekteissa pyritään työn aikana välttämään vanhentuvia komentoja ja ominaisuuksia molemmissa kehitysympäristöissä, mutta oletettavasti jotkin käytettävät metodit saattavat vanhentua tulevaisuudessa.

Sovelluksen testaaminen eri laitteilla on tehty Expolla helpoksi. Expon avatessa projektin komentokehotteeseen ilmestyy projektikohtainen QR-koodi, jonka skannaamalla ilmaisessa Expo-mobiilisovelluksessa projektin mobiilisovellus aukeaa laitteeseen. Jokaista kehityksessä olevaa sovellusta ei siis tarvitse erikseen asentaa testilaitteeseen, mikä säästää testilaitteen muistia. Expon testisovellusmetodi

on erittäin hyvä iOS-laitteilla testaamiseen, sillä vaikka sovelluskehitys tapahtuisi Windows-laitteella, sovelluksen voi silti testata iOS-laitteella QR-koodin avulla. Expo ei kuitenkaan tarjoa suoraan emulaattorilaitteita testaukseen, vaan ne on luotava Android Studion emulaattorien kautta. Windows-laitteella sovelluskehitykseen ei voi saada suoraan iOS-emulaattorilaitetta testaukseen, sillä iOS-emulaattori vaatii Apple ID:n, jonka täsmäämisen tarkistamiseen tarvitaan Mac-laite. (Expo Team, 2020.) Expon sovelluskehitykseen tarvitaan iOS-laite, ja tarvittaessa emuloitu Android-laite, jos sovelluskehitystä ei tehdä Mac-laitteella.

Flutterilla Android-laitteiden testaus vaatii ajuriohjelmiston asentamisen, mobiililaitteessa USB-virhehallinnan käyttöönoton, ja mobiililaitteen USB:n liittämisen kehitystietokoneeseen. Tämän jälkeen kehityksen alla olevan sovelluksen voi ajaa testilaitteeseen suoraan Android Studiosta. (Flutter, 2020.) Expon kaltaista testausta varten tehtyä mobiilisovellusalustaa ei ole. Alustariippumaton sovelluskehitys Flutterilla onkin yksinkertaisinta Mac-laitteella, sillä iOS-laitteella Flutter-sovellus avautuu testattavaksi vain, jos testimobiililaitteen ja kehitystietokoneen Apple ID:t täsmäävät, ja iOS-emulaattorisovelluksia voi luoda Android Studion emulointityökalulla. Windows-laitteella vain Android-laitteeseen voi ajaa testisovelluksen, ja emulaattorilla voi luoda vain Android-sovelluksia. Flutterin iOS-testaukseen Windows-laitteella on tarjolla monia työkaluja sovelluskehityksen helpottamiseksi, esimerkiksi simulaattorit ja virtuaalikoneet, joilla testaamisen voi suorittaa. Flutterin saumaton testaus on Expoon verrattuna rajoitetumpi, ja vaatii enemmän työtä iOS-testaukseen.

Windows-laitteella sovelluskehityksen iOS-testaamiseen Expolla ja varsinkin Flutterilla on erilaisia työkaluja, joiden käyttöön on hyvä tutustua. Testaukseen käytettävään iOS-laiteeseen voi esimerkiksi suorittaa Jailbreakin, eli kolmansien osapuolien ohjelmistojen sallimisen iOS-käyttöjärjestelmässä (Azad ym. 2020). Tällaisia työkaluja ovat esimerkiksi Codemagic Apple-sertifikaatin automaattiseksi luomiseksi ilman Mac-laitetta (Nevercode, 2020) ja Cydia Impactor App Storen ulkopuolisen sovelluksen asentamiseksi (Saurik, 2020). Varsinkin Flutterin valitsemisessa on otettava huomioon käyttöjärjestelmä, jolla sovelluskehitys tullaan suorittamaan. Expo on sovellusten testaamisen näkökulmasta helppokäyttöisempi. Testisovellusten kehityksessä oli käytössä Jailbreak iOS-laite, jolla pystyi testaamaan molemmat sovellukset myös iOS-laitteella, mutta iOS-testaus Windows-kehityslaitteella ei ole helppoa, varsinkaan aloittelijalle.

Valmiiden sovellusten lisäkehitys on mutkattomampaa Expolla, koska jo julkaistujen sovellusten muutokset päivittyvät Playstoreen sekä App Storeen reaaliajassa (Expo Team, 2020). Julkaistujen Flutter-

sovellusten päivittymisessä mobiilisovelluskauppoihin on vaihteleva viive (Flutter, 2020a). Expo-sovelluksiin on siis helpompaa tarjota nopeita korjauksia ja säännöllisiä päivityksiä, kun taas Flutter-sovellusten käyttäjän täytyy odottaa päivityksiä.

Flutterin ja Reactin käyttöönotossa oli paljon samankaltaisuuksia. Toisen kehitysympäristön ollessa heikompi tietyllä alueella, toinen oli vahvempi. Esimerkiksi vaikka React Expon asennus oli monimutkaisempaa, sovellusten testaaminen eri laitteilla oli huomattavasti helpompaa Flutteriin verrattuna. Alustan valitsemisessa on hyvä ottaa huomioon, kumman huonot puolet on yrityksessä helpompi käsitellä ja kumman hyvät puolet ovat hyödyllisempiä projektin kannalta.

5 SOVELTUVUUS ERILAISIIN YRITYKSIIN

React Expon tai Flutterin valinta on vahvasti sidonnainen yrityksessä työskentelevien sovelluskehittäjien aiempiin taitoihin (Ravichandran, 2019). Entä jos aiemmat kokemukset on vaikeaa yhdistää kummankaan alustan ohjelmointikielen opetteluun? Entä jos aiempaa kokemusta ei ole merkittävästi? Työssä pyritään kokemuspohjalta testaamaan, kumpi kehitysalusta soveltuisi paremmin ja millaisille yrityksille.

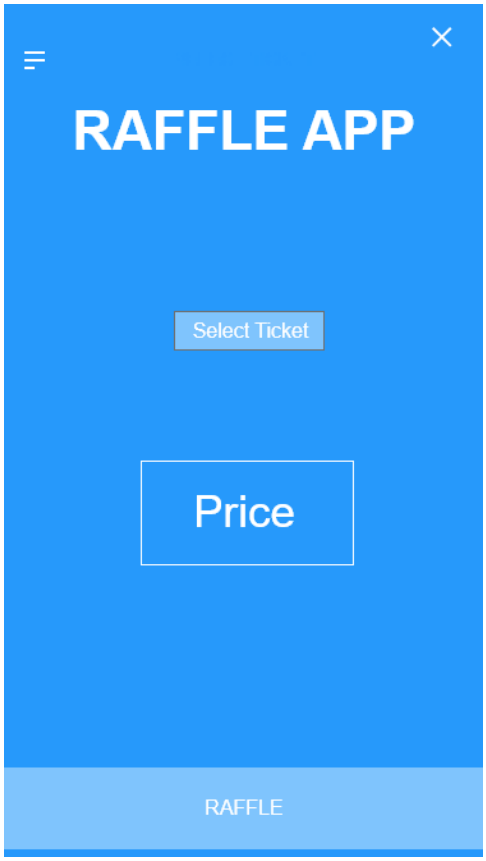
Molemmissa kehitysympäristöissä luotiin yksi sovellus, eli yhteensä kaksi testisovellusta. Molempien sovellusten pääperiaatteena oli toimia arpakoneena, joka arpoo, millaisen palkinnon arvan ostanut käyttäjä saa. Sovellusten luomisen pääperiaatteena oli kehitysympäristöjen toimintaan syventyminen ja sovelluskehitysprosessin vertailu kehitysympäristöjen välillä. Testisovelluksista pyrittiin ohjelmoimaan mahdollisimman aloittelijaystävälliset sovelluskehittäjän näkökulmasta esimerkkinä siitä, mitä eri yritykset voivat kartoittaa omissa sovelluskehittäjätiimeissään kehitysalustaa valittaessa.

Mobiilisovelluskehitys eroaa pääasiassa tietokoneella käytettävistä sovelluksista sekä kosketusnäytöille suunnatussa ulkoasussa että prosessointikyvyn säästämisessä. Sovelluskehityksessä pyritään ottamaan huomioon mobiilialustojen rajoitetut resurssit, pitämään sovelluksen ulkoasu mahdollisimman yksinkertaisena sekä pitämään sovellus kokonaisuudessaan helppokäyttöisenä, nopeana ja kätevänä. (IBM Corporation, 2011.) Sovelluskehitysmenetelmät ja etiikka pyrittiin pitämään samana molemmilla alustoilla erojen löytämisen helpottamiseksi ja selkeyttämiseksi, vaikka koodin rakenteessa tulee väistämättä olemaan eroja ohjelmointikielen mukaan.

Sovellukset pyrittiin yksinkertaisina, jotta kehityserot olisi helpompi tunnistaa, eikä sovelluskehitys seisahtuisi esimerkiksi tietynlaisten, monimutkaisten työkalujen ja metodien etsimiseen. Koodauksen aikana pitäydyttiin niin joustavissa ja kestävissä ohjelmointimetodeissa kuin mahdollista. Erillisten sivujen ulkoasun selkeyden lisäksi sovelluksen tulisi olla kokonaisuudessaan dynaaminen, jotta sen ulkoasu pysyisi mahdollisimman samankaltaisena ja käyttöystävällisenä eri käyttöjärjestelmissä, näyttökoissa sekä vaaka- ja pystysuorassa. Sovelluksen on myös hyvä olla nopea käyttömukavuuden maksimoimiseksi, joten prosessointikykyä säästäviä koodausmenetelmiä olisi hyvä käyttää. (IBM Corporation, 2011.) Sovelluskehityksessä pyrittiin pitäytymään näissä koodausperiaatteissa molemmissa sovel-

luskehitysympäristöissä. Yrityksen olisi hyvä pystyä pitäytymään parhaaksi valitsemassaan suunnitelmassa ja valitsemassaan etiikassa sovelluskehitysalustasta huolimatta, mutta huomioida, millaisia eroja ja ongelmakohtia tietyn alustan valitseminen voi aiheuttaa.

Molempien testisovellusten suunnitteluun käytettiin samaa mallia. Sovelluksen etusivulla näkyy sovelluksen nimi, tapa valita lipuke palkintovaihtoehtojen määrittämiseksi, arvontapainike, tila palkinnon näyttämiselle, tapa siirtyä sovelluksen asetuksiin ja mahdollisuus sulkea sovellus (KUVA 1). Etusivun tarkoituksena on olla yksinkertainen, helppokäyttöinen ja selkeä käyttömukavuuden maksimoimiseksi.



KUVA 1. Testisovellusten etusivun ulkoasun mallikuva.

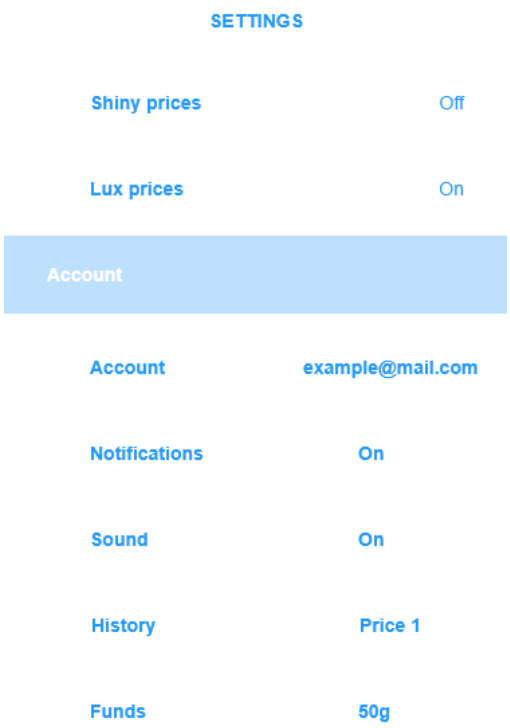
Sovelluksen toisella sivulla, joka aukeaa etusivun lipunvalintapainikkeen avulla, voi valita lipukevaihtoehdon, jonka perusteella sovellus arpoo etusivulla palkinnon. Eri vaihtoehdoilla on eri hinta, ja palkinnon arvo on riippuvainen lipukkeen hinnasta. Toiselta sivulta pääsee etusivulle painamalla mobiili-

laitteen edellinen painiketta tai valitsemalla lipukevaihtoehdon. Toiselta sivulta pääsee myös sovelluksen asetuksiin. (KUVA 2.) Toisen sivun tarkoituksena on selkeästi esittää käyttäjälle kyseessä olevan lipukevalintasivu otsikon sekä selkeiden lipukevaihtoehtojen avulla, ja jokaisen lipukkeen hinnan on oltava esillä selkeästi. Asetussivun avaamiseksi olisi hyvä käyttää samanlaista kuvaketta kuin etusivulla helpon käytön mahdollistamiseksi.

KUVA 2. Testisovellusten toisen sivun ulkoasun mallikuva.

Sovelluksen viimeinen sivu toimii käyttäjäasetussivuna. Asetussivulla käyttäjä voi valita tietynlaisten palkintojen arpamahdollisuuden, sovelluksen ääniasetukset ja käyttäjän tilillä olevan leikkirahan määrän. Asetuksista näkyy myös käyttäjän edellinen voitto. (KUVA 3.) Asetussivulla käyttäjä voi esimerkiksi poistaa käytöstä erilaiset palkintoversiot, vaikkapa tietynlaisen palkinnon kiiltävän version. Ase-

tussivun tarkoitus on olla käyttäjäystävällinen selkeällä ulkoasulla ja tarjota minimaalinen määrä asetusvaihtoehtoja käyttöystävällisyyden maksimoimiseksi. Esimerkkisovelluksissa pidettiin asetussivu vain ulkoasuna, eikä siinä voi siis tehdä muutoksia sovelluksen toimintaan.



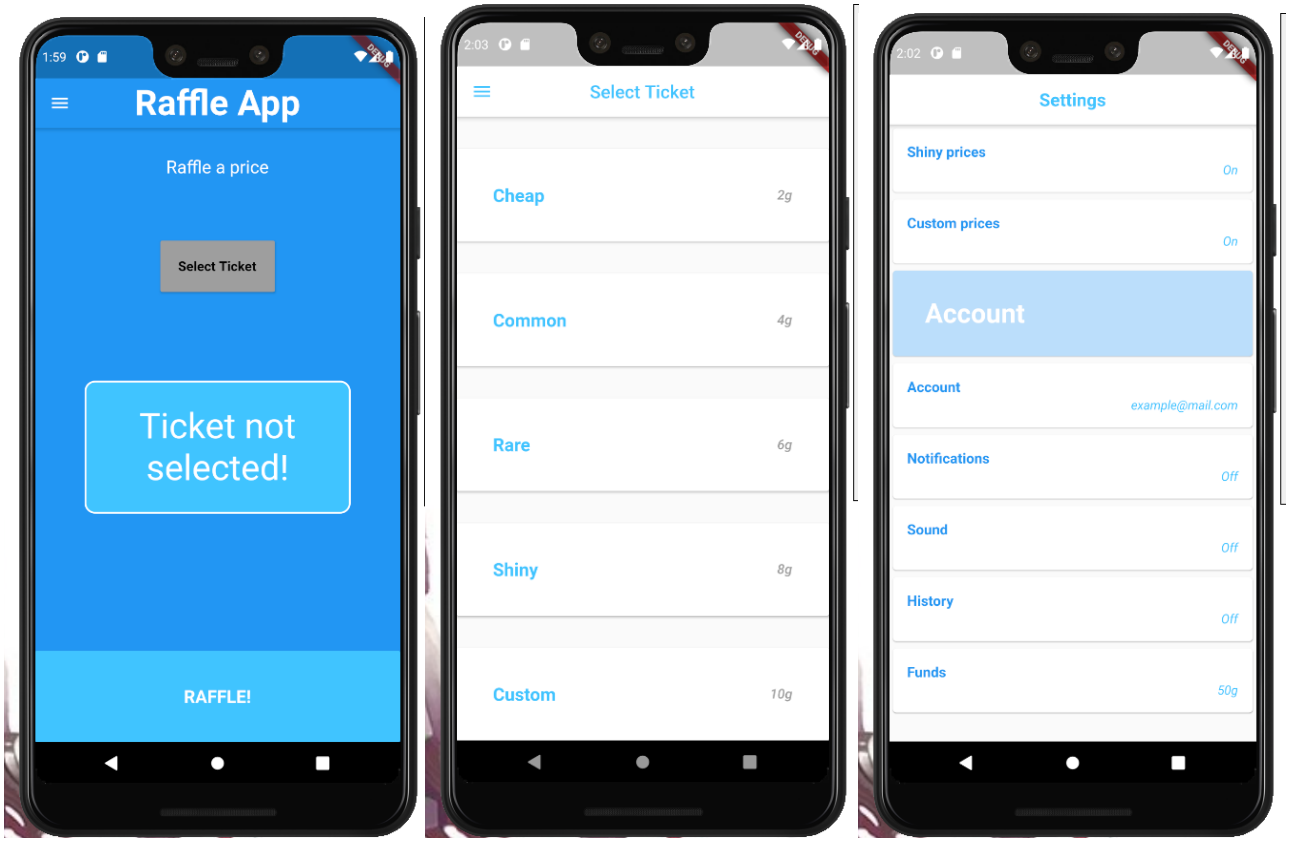
KUVA 3. Testisovellusten kolmannen sivun ulkoasun mallikuva.

Sovelluskehityksessä pyrittiin pitäytymään mallikuvissa (KUVA 1, KUVA 2 ja KUVA 3) mahdollisimman paljon. Suunnitelmaa seuraamalla yritettiin nähdä, kuinka samannäköiset sovellukset voi kehittää eri kehitysalustoilla visuaalisesti ja millaisia eroja se vaatii koodaustasolla. Kuvamallien arveltiin olevan aloittelevan yrityksen ensimmäinen suunnitelma UML-kaavion lisäksi. Ulkoasun ollessa yksinkertainen, sovellusten ulkoasujen oletetaan olevan samankaltaiset, mutta täysin mallissa pitäytyminen on vaikeaa. Sovellusten ulkoasujen pyrittiin siis olevan mallikuvista tunnistettavasti inspiroituja. Visuaalisesti monimutkaisemmissa sovelluksissa samankaltaisuuden saavuttamisen oletetaan olevan vaikeampaa visuaalisten komponenttien kirjastojen erilaisuuden takia.

5.1 Flutterin sovelluskehityskokemus

Sovelluskehitykseen käytettiin Flutteria Android Studiolla dokumentaation suositusten mukaisesti. Android Studio on Flutterin oletuskehitysalusta, ja se takaa parhaimmat päivitysmahdollisuudet, joten Flutterin käyttö toisella ohjelmointialustalla voi olla vaikeaa. Esimerkiksi versiopäivitykset ja pakettien käyttöönotto on helpompaa Android Studion automatisoidun prosessin vuoksi verrattuna esimerkiksi Visual Studioon, jossa versiopäivitykset on määritettävä manuaalisesti. (Flutter, 2020a) Android Studio on ilmainen, joten sen optimaalisuuden voi hyödyntää esimerkiksi käyttämällä sitä toisen alustan ohella.

Android Studion aiemman käyttökokemuksen ja Flutterin tarjoamien ohjeiden avulla Dartin oppiminen ohjelmointikielenä sekä Flutterin käytön hallitseminen kokonaisuudessaan sujui nopeammin kuin Expon. Dart oli ohjelmointikielenä miellyttävä ja se sai nopeammin käyttömukavuutta kuin Expon JavaScript. Widgetit olivat loogisempia kuin Expon ulkoasujen rakentaminen, vaikka niissä olikin paljon samankaltaisuuksia. Uusien kirjastojen käyttöönotto oli helppoa, ja sovelluksesta sai visuaalisen käyttöversion todella nopeasti. Dartin käyttöönotto oli helppoa, sillä se muistutti paljon Javaa, jota on käytetty Android Studiolla ennen työn aloittamista. Dart tuntui dynaamisemmalta versiolta Javasta, joka toimi Javascriptin logiikalla. Dartin suurin heikkous on se, ettei sitä oikeastaan tarvita Flutterin käytön ulkopuolella. Uuden ohjelmointikielen opiskeltua käytön haluaisi olevan mahdollisimman universaali.



KUVA 4. Flutter-sovelluksen ulkoasu eri sovellussivuilla.

Flutterin sovelluskehityksen aikana oli helppoa pitäytyä mallikuvissa (KUVA 1, KUVA 2 ja KUVA 3). Mallikuvan mukaisten ulkoasukomponenttien löytäminen oli helppoa, niiden muokkaaminen oli dynaamista, ja Flutterilla hyvin mallia jäljittelevän lopputuloksen (KUVA 4). Todennäköisesti Flutterilla olisi voinut luoda täysin mallikuvien kanssa identtisen sovelluksen, mutta emulaattorikuvissa näkyvä lopputulos (KUVA 4) oli tarpeeksi tyydyttävä.

Flutteria käyttäessä sai Expoa nopeammin UI-komponentteja käyttöön, ja kehittäjä pääsi todella nopeasti muokkaamaan sovelluksen ulkoasua. Flutter soveltuu todella hyvin MVP-sovellusten kehitykseen, sekä nopeaan tapaan kehittää prototyyppisovellus asiakkaan testattavaksi. Prototyyppisovelluksen sai alkuun todella nopeasti, mutta muiden ominaisuuksien käyttöönotto vei paljon aikaa, ja suurin osa prototyyppikoodista oli kirjoitettava uudelleen. Nopeiden prototyyppien pohjalta sovelluksen rakentaminen ei ole siksi välttämättä kannattavaa, ellei ulkonäön haluta vastaavan prototyyppiä täysin. Mallikuvan pohjalta sovelluksen rakentaminen oli kuitenkin nopeampaa ja helpompaa kuin Expossa.

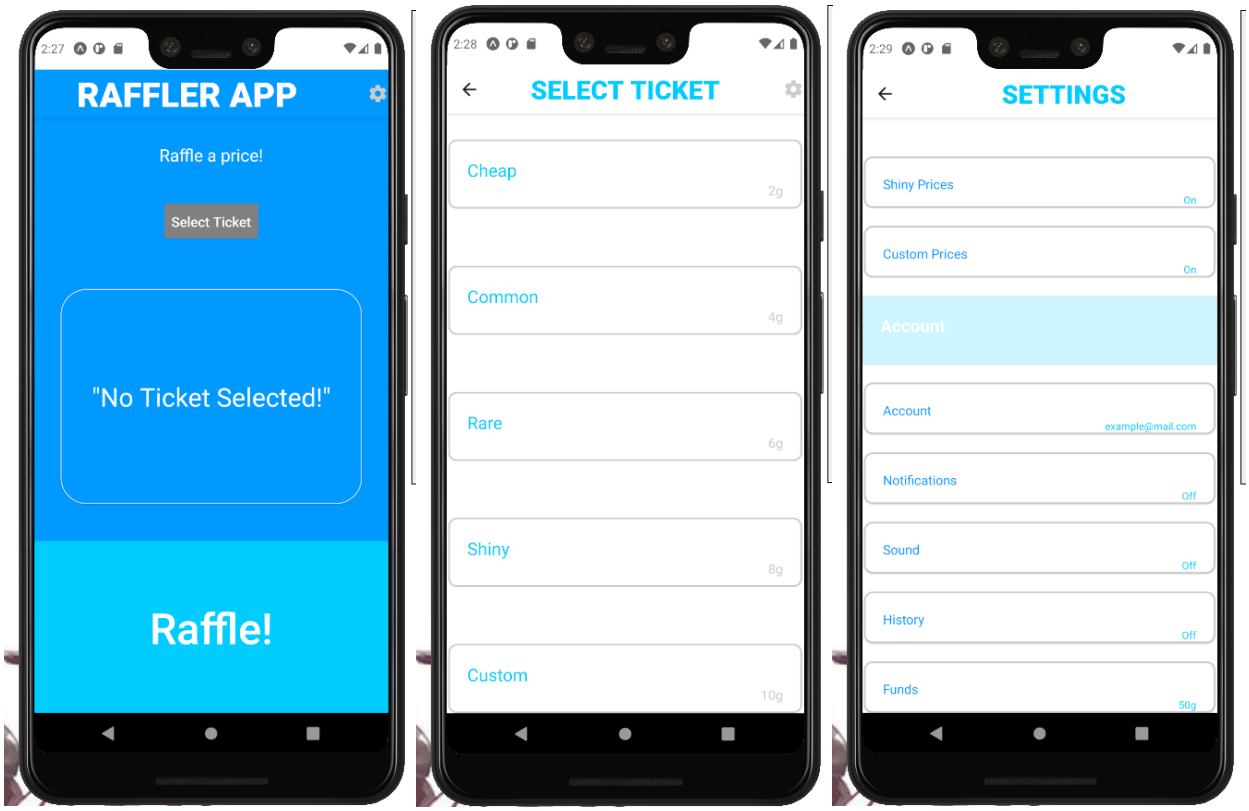
Uusien ominaisuuspakettien käyttöönotto oli Flutterissa helppoa ja luotettavampaa kuin Expossa. Paketti asennetaan pubspec.yaml tiedoston avulla, jonka jälkeen se asennetaan Flutterin komentokehoteikkunassa. Android Studio hoiti asentamisen kokonaan, kun oikeanlainen viittaus pakettiin oli lisätty tiedostoon, joten esimerkiksi käyttäjäkohtaiset virheet oli minimoitu ja ne olivat helposti löydettävissä yhdessä tiedostossa.

5.2 React Expon sovelluskehityskokemus

Expon asennus ja käyttöönotto olivat helppoa, mutta React Expon kehitysympäristön käytön opettelu Visual Studio Codella oli haastavaa. Työkalujen, metodien ja ohjeiden löytäminen oli myös Flutteriin verrattuna vaikeampaa. Oli esimerkiksi vaikeaa löytää, mitkä React Nativen työkalut olivat toiminnassa Expossa ja mitkä eivät.

JavaScript on yleisesti tunnettu ohjelmointikieli, mutta ennen työn aloittamista sitä ei ollut käytetty aktiivisesti muissa ohjelmointiprojekteissa, minkä takia JavaScript koodirakenteen ymmärtämisessä kesti pidempään kuin Dartin. Olio-ohjelmointi on ennen työn aloittamista harjaantunut koodausperiaate, ja sen puute tuotti vaikeuksia. Testisovelluksen ohjelmoinnin aikana kesti enemmän aikaa saada luotettava käyttömukavuuskokemus JavaScriptiin.

Expon sovelluskehityksen aikana oli paljon ongelmia ymmärtää, mitkä ongelmat sovelluksessa johtuivat mistäkin koodiriveistä. Expo-sovelluksessa kokeiltiin erilaisia funktioita ja luokkia, kunnes lopulta saatiin haluttuja tuloksia esille. Kesti oletettua pidempään oppia monet JavaScriptin ongelmakohdat, esimerkiksi OnPress-komennoissa kutsuttujen funktioiden loppumattomat päivityskierteet koettiin epäselviksi, ja etsittiin pitkään tietoa siitä, miten ne saisi korjattua. Visual Studio Code ei ilmoittanut monista ongelmakohdista, ja välillä Expon kokoaja eikä myöskään Expo Client kertonut, mikä sovelluksessa oli pielessä. Flutteriin verrattuna ongelmien löytäminen ja korjaaminen oli haastavampaa, mutta oletettavasti eri sovelluskehitysympäristössä ongelmakohtien ratkaiseminen olisi saattanut olla helpompaa.



KUVA 5. Expo-sovelluksen ulkoasu eri sovellussivuilla.

Expossa ulkoasun jäljittely mallikuvasta (KUVA 1, KUVA 2 ja KUVA 3) ei ollut yhtä yksinkertaista kuin Flutterissa. Esimerkiksi jouduttiin etsimään todella kauan eri UI-komponentteja, joilla saataisiin mahdollisimman samankaltainen lopputulos kuin mallikuvissa. Sovellussivut (KUVA 5) ovat tunnistettavissa mallikuvien inspiroimiksi, mutta vielä tämän projektin aikana saaduilla Expon käyttötaidoilla ei koettu mahdolliseksi päästä emulaattorikuvia lähemmäs mallia. Kehittäjän näkökulmasta olisi toivottu esimerkiksi asetussivusta paremman näköistä. Mallikuvan suora jäljittely oli helpompaa Flutterilla, joten kenties Expo-kehitykseen soveltuisivat paremmin esimerkiksi UML-kaaviot.

Eri näkymien rakentaminen Expossa oli miellyttävää, vaikka sen hallitseminen vei aikaa. Flutterin näkymien rakentaminen oli loogisempaa, mutta Expossa näkymien rakentaminen ja muokkaus tuntuivat yksinkertaisemmilta. Expossa oli paljon selkeämpää, mihin kohtaan minkäkin komponentin ulkonäköominaisuudet lueteltiin. Expossa pystyi luomaan oletusulkonäköasetukset helposti StyleSheet-työkalulla kullekin sovellussivulle, ja poikkeukset oli helppoa määrittää kyseisen komponentin kohdalla tarvittaessa. Flutteria käyttäessä ei käytetty tai otettu selvää samankaltaisen ominaisuuden olemassaolosta tai sen käyttömahdollisuuksista.

Expon tarjoamalla mobiilisovelluksella käyttäjä voi testata projektiaan asentamatta sovellusta testilaitteelle. Kun Expo-sovelluksen avaa NPM:n avulla, komentokehotteeseen ilmestyy QR-koodi, jonka lukemalla se avautuu mobiililaitteeseen. Sovelluksen voi avata testilaitteeseen erilaisilla yhteysmetodeilla, esimerkiksi luomalla paikallisen QR-koodin kehityksessä käytetyn tietokoneen kanssa yhdistettyyn laitteeseen. Sovelluksia voi testata sekä iOS- että Android-laitteilla sovelluksen kautta, mikä helpottaa ja nopeuttaa testausta.

NPM pakettihallintaohjelman käyttö aiheutti ongelmia sovelluskehityksessä. Ohjeiden mukaisesti projektiin asennettiin NPM:n avulla kirjastoja, joista saatiin sovellukseen uusia ominaisuuksia käyttöön, mutta NPM ei ilmoita, jos paketin asennuksessa tapahtuu poikkeus tai se asentaa ominaisuuksia edellisten päälle. Työn aikana ei käytetty toista Expon tukemaa pakettienhallintatyökalua Yarnia, mutta oletettavasti samat ongelmat ovat läsnä myös sen käytössä. Pakettien tarjoamien työkalujen käyttöönoton toivottaisiin muuttuvan vähemmän riskialttiiksi.

Expon käyttö Flutteriin verrattuna oli hyvin samankaltaista, ja näkymien rakentaminen oli miellyttävää, mutta sovelluskehityksessä koettiin paljon enemmän ongelmia Expon käytön aikana kuin Flutterin. Expon ongelmien ymmärtäminen, etsintä ja korjaaminen oli aloittelijan näkökulmasta työläämpää kuin Flutterissa. Jos paketinhallintatyökaluna yrityksessä käytetään NPM-työkaluja, sovelluskehitys voi hidastua jo pelkästään pakettien asennuksista johtuvien ongelmien takia. Expo kuitenkin vaatii vähemmän prosessointivoimaa sovelluskehitykseen, sovelluksia on helpompi testata eri laitteilla ja se ei ole vahvasti sidonnainen vain yhteen sovelluskehitysalustaan, kuten Flutter on Android Studioon.

5.3 Yhteenveto

Sovelluskehityksen loputtua molemmilla alustoilla Flutterin käyttö oli loppujen lopuksi miellyttävämpää. Flutterilla sai nopeammin visuaalisia lopputuloksia, eri pakettien ominaisuuksien käyttö ja käyttöönotto oli helppoa ja saadut lopputulokset olivat miellyttäviä. Expon testialusta on sen sijaan huomattavasti parempi kuin Flutterin, mikä voi pidemmällä aikavälillä olla hyödyllisempää. Molemmilla alustoilla on hyvät ja huonot puolensa, jotka on hyvä vielä eritellä.

Koska ennen työn aloittamista oli eniten ohjelmointitaustaa C++-, Java- ja C#-kielillä, Flutteria suositellaan yrityksille, jonka ohjelmoijat ovat keskittyneet kyseisten ja samankaltaisten kielien käyttöön.

Sen sijaan yritykset, joiden ohjelmointitiimillä on paljon kokemusta verkko-ohjelmoinnissa, varsinkin JavaScriptin ja sen kanssa samankaltaisten ohjelmointikielien käytöstä, React Expo on hyvä valinta ensimmäiseksi mobiilisovellusten kehitysalustaksi.

React Expo on parempi valinta, jos yrityksessä on rajoitettu määrä testilaitteita ja kehitykseen käytettäviä tietokoneita, sillä Flutterin testaus iOS-laitteilla on monimutkaisempaa kuin Expon. Jos yritys tarvitsee paljon nopeita prototyyppisovelluksia, testaus virtuaalilaitteilla riittää tai ohjelmoijatiimi on valmis työskentelemään iOS-ympäristön kanssa enemmän, Flutter on sen aloittelijaystävällisyyden ja nopeiden visuaalisten tulosten takia parempi valinta. Jos kehitysaikataulu ei ole tiukka, Flutterin iOS-testauksen voi suorittaa myös Play- ja App Storen betatestaus-toiminnolla, jolloin sovelluksen saa käyttöön iOS-laitteelle ilman kiertoteitä.

Flutter on Android Studiossa käytettäessä optimaalisin, ja myös rajoitetuin. Jos yrityksellä on resurssit pitää Android Studiota kehitystyökaluna tai sen lisäosana, Flutter on hyvä valinta. React Expoa voi käyttää ilmankin Android Studiota, mutta sen virtuaalisten emulaattoreiden käyttö vaatii Android Studion työkaluja, joten React Expon käytössä sen asennus saattaa olla tarvittavaa. Lisäksi React Expon ongelmien löytäminen ja korjaaminen voi olla vaikeaa kehitysalustan mukaan, ja sopivan alustan löytäminen voi viedä aikaa.

React Expon ja Flutterin sovellusten suorituskyky pienemmissä sovelluksissa ei eroa käyttäjän näkökulmasta paljon, mutta raskaammissa projekteissa Flutter tarjoaa paremman suorituskyvyn ja lyhyemmät latausajat (Mroczkowska ym. 2019). React Expo sopii paremmin projekteihin, jotka eivät vaadi korkeaa suorituskykyä tai pitkät latausajat eivät ole ongelma.

Flutterin käyttämä Dart on helppo oppia tietyllä ohjelmointitaustalla, mutta sille ei ole käyttötarvetta Flutter-kehityksen ulkopuolella. Sen opiskelu voi sen takia olla laajasta näkökulmasta epäkannattavaa, ellei yritys vaadi pidempiaikaista Flutter-kehitystä ja sovellusten ylläpitoa. Ohjelmointikielen näkökulmasta React Expo voi olla parempi vaihtoehto, sillä JavaScript ohjelmointikielenä on laajemmin käytössä muillakin alustoilla.

Myös sovelluskehitysalustan tulevaisuus on otettava huomioon. Jos yritys valitsee esimerkiksi Flutterin, mutta sen tuki ja kehitys lopetetaan, ylimääräisiä kuluja voi kertyä huomattavasti. Flutteria ja React Nativen perhettä ylläpitää isot yritykset, Google ja Facebook, ja molemmilla on laaja kehittäjäkommuuni, joka tarjoaa päivitettyjä ominaisuuksia ja korjausehdotuksia. React Native on Flutteria

vanhempi, mutta React Expo aloittelijatyökaluna on vapaaehtoisten ylläpitämä. (Facebook Open Source, 2020; Flutter, 2020b) Sen tulevaisuus on siis epävakaampi Flutteriin verrattuna, mutta React Native kokonaisuudessaan on joko yhtä luotettava tulevaisuuden kannalta, ellei jopa luotettavampi.

Kriteeri	**Flutter**	**React Expo**
Kehittäjän ohjelmointitausta	C, C++, C#, Java ja samankaltaiset kielet.	HTML, XML, Python ja samankaltaiset kielet.
Testimahdollisuudet	Rajoitetut testimahdollisuudet. Ongelmia varsinkin iOS-testauksessa.	Laajat testimahdollisuudet Expo-mobiilisovelluksen avulla.
Ohjelmointiympäristö	Rajoitettu. Android Studio optimaalisin.	Laaja, mutta ongelmien kartoitus vaikeaa eri ympäristöissä.
Suorituskyky	Hyvä. Natiivin suorituskyvyn hyödyntäminen takaa nopean sovelluksen.	Huono. Raskaissa sovelluksissa huomattavasti pidemmät latausajat.
Ohjelmointikieli	Dart. Vain Flutterissa käytössä oleva kieli.	JavaScript. React Nativen ulkopuolella käytössä oleva kieli.
Alustan ylläpito	Google. Iso yritys ja aktiivinen kommuuni.	Facebook. Iso yritys ja aktiivinen kommuuni.

TAULUKKO 1. Flutterin ja React Expon vertailu tiivistettynä.

Taulukon (1) perusteella Flutter on selvästi rajoitetumpi kuin Expo. Sovelluskehityksen jälkeen on kuitenkin selkeämpää, että mobiilisovelluskehitystä aloittelevalle yritykselle Flutter on parempi valinta, jos tiimillä ei ole ohjelmointitaustaa tai sitä ei osata suoraan yhdistää kumpaankaan ohjelmointikieleen. Dartin opetteluun oli selkeät ohjeet Flutterin kehitysohjeessa (Flutter, 2020a), ongelmien löytäminen ja korjaaminen oli helpompaa ja nopeat visuaaliset tulokset helpottavat oppimista. Jos yrityksellä ei ole resursseja Flutterin iOS-testausongelmien korjaamiseen ja iOS-testaus kehityksen aikana on välttämätöntä, sekä ohjelmointitiimin taidot ovat selvästi JavaScriptin puolella, Expo on parempi valinta. Suurimmalta osalta Flutter on kuitenkin parempi valinta, sillä Expon käytöstä kuitenkin siirrytään muihin

React Native -perheen työkaluihin, ja kolmansien osapuolien työkaluja vaaditaan huomattavasti enemmän. Flutter sopii hyvin pienelle tiimille ja yksinkertaiselle yrityssovellukselle, eikä sen opetteluun uppoa liikaa resursseja yritysjohdon näkökulmasta.

Valinta React Expon ja Flutterin kohdalla on hyvin sidottu sovelluskehittäjätiimin kokoonpanoon, yrityksen resursseihin ja projektin tarpeisiin (Mroczkowska, Skuza & Wlodarczyk, 2019). Jos yrityksessä ei löydy selkeää parempaa vaihtoehtoa hyvien ja huonojen puolien erittelyn jälkeen, Flutter on varmempi valinta. Jos yrityksen olisi parempi valita React Expo, syiden tulisi olla selkeästi Expon puolella projektin toteuttamisen ja ylläpidon näkökulmasta.

LÄHTEET

Azad, B., Bednarz, D., Bingner, S., exDeveloper, James, J., PhoneRebel, pwn20wnd, Siguza, Ubik & Wiliamson, N. 2020. unc0ver: The most advanced jailbreak tool. unc0ver. Saatavissa: https://unc0ver.dev/. Viitattu 30.11.2020.

Bura, D., Cajthaml, O., Hasa, F., Kraft, L. & Vavra, M. 2020. The ultimate guide to mobile app development. Pixelfield. Saatavissa: https://pixelfield.co.uk/app-development/. Viitattu 6.4.2020.

Expo Team. 2020. Introduction to Expo. Facebook Inc. Saatavissa: https://docs.expo.io/. Viitattu 21.11.2020.

Facebook Open Source. 2020. React Native Docs. Facebook Inc. Saatavissa: https://reactnative.dev/docs/getting-started.html. Viitattu 24.11.2020.

Flutter. 2020a. Flutter documentation. Google. Saatavissa: https://flutter.dev/docs. Viitattu 24.11.2020.

Flutter. 2020b. Flutter. Google. Saatavissa: https://flutter.dev/. Viitattu 24.11.2020.

Galadzhii, A. 2020. Best Cross-Platform Mobile Development Tools in 2020. LITSLINK. Saatavissa: https://litslink.com/blog/best-cross-platform-mobile-development-tools-in-2020. Viitattu 23.7.2020.

Gupta, A. 2019. Top Absolutely Necessary Mobile App Development Tools and Platforms of 2020 To know. SAG IPL. Saatavissa: https://blog.sagipl.com/mobile-app-development-tools/. Viitattu 21.11.2020.

IBM Cloud Education. 2018. Mobile Application Development. IBM. Saatavissa: https://www.ibm.com/cloud/learn/mobile-application-development-explained. Viitattu 21.11.2020.

IBM Corporation. 2011. IBM Cúram Social Program Management, Version 7.0.2, 7.0.3 Refresh Pack, and 7.0.4 Refresh Pack. IBM. Saatavissa: https://www.ibm.com/support/knowledgecenter/SS8S5A_7.0.4/com.ibm.curam.content.doc/RestfulAPI/BusinessAnalysis/c_RAPI_BA_Defining-MobileAppPageRequirements.html. Viitattu 6.4.2020.

Kielitoimiston sanakirja. 2020. Helsinki: Kotimaisten kielten keskuksen verkkojulkaisija 35. Saatavissa: https://www.kielitoimistonsanakirja.fi/#/mobiilisovellus. Viitattu 6.4.2020.

Mroczkowska, A., Skuza, B. & Wlodarczyk, D. 2019. Flutter vs. React Native – What to Choose in 2020? Droids On Roids. Saatavissa: https://www.thedroidsonroids.com/blog/flutter-vs-react-native-what-to-choose-in-2020. Viitattu 23.7.2020.

Muller, H. 2020. FAQ. Google. Saatavissa: https://flutter.dev/docs/resources/faq. Viitattu 24.11.2020.

Nader, Y. 2020. React Native vs Flutter. hackr.io. Saatavissa: https://hackr.io/blog/react-native-vs-flutter. Viitattu 23.7.2020.

Nevercode. 2020. Codemagic: Build, test and deliver mobile apps in record time. Nevercode Ltd. Saatavissa: https://codemagic.io/start/. Viitattu 30.11.2020.

Ravichandran, A. 2019. React Native or Flutter – What Should I Pick To Build My Mobile App? Medium. Saatavissa: https://medium.com/@adhithiravi/react-native-vs-flutter-what-are-the-differences-b6dc892f0d34. Viitattu 23.7.2020.

Saurik. 2020. Cydia Impactor. Saatavissa: http://www.cydiaimpactor.com/. Viitattu: 23.11.2020.

Zhang, J., Sagar, S. & Shihab, E. 2013. The evolution of mobile apps: An exploratory study. 1st International Workshop on Software Development Lifecycle for Mobile, DeMobile 2013 - Proceedings. 1-8. 10.1145/2501553.2501554. Saatavissa: https://www.researchgate.net/publication/262332106_The_evolution_of_mobile_apps_An_exploratory_study. Viitattu 23.7.2020.

Raffler Expo

```
import { StatusBar } from 'expo-status-bar';
import 'react-native-gesture-handler';
import React, { Component} from 'react';
import { StyleSheet, Text, View, Dimensions, TouchableOpacity } from 'react-na-
tive';
import { NavigationContainer} from '@react-navigation/native';
import { createStackNavigator } from '@react-navigation/stack';
import { Card, Button, Icon} from 'react-native-elements';

const Stack = createStackNavigator();

widthd = Dimensions.get('window').width;

//Main Page
class MainScreen extends Component {
  constructor(){
    super();
    this.state={
      click: false,
      price: 'none',
    }
  }

  cheapArray = [
    'Red Fox',
    'Cross Fox',
    'Silver Fox',
    'Pearl Fox',
    'Naali Winter',
  ]

  commonArray = [
    'Burgundy Fox',
    'Amber Fox',
    'Shadow Fox',
    'Ice Fox',
    'Naali Summer',
  ]

  rareArray = [
    'Platinum Fox',
    'Black Cross Fox',
    'Gold Fox',
    'Platinum Cross Fox',
    'Naali Autumn',
  ]

  shinyArray = [
    'Red Marble Fox',
    'Marble Fox',
    'Burgundy Marble Fox',
    'Cross Marble Fox',
    'Naali Spring',
  ]

  customArray = [
```

```
  'Custom Fox',
  'Naali Custom',
]

textSet = (text) => {
  if(this.state.click == true){
    return this.state.price;
  }
  else{
  return JSON.stringify(text);
  }
}

rndSelection = (option) =>{
  var op = JSON.stringify(option);
  var nr
  if(op == "\"Cheap\""){
     nr = Math.floor(Math.random()*this.cheapArray.length);
     this.setState({price: this.cheapArray[nr]});
  }
  else if(op == "\"Common\""){
    nr = Math.floor(Math.random()*this.commonArray.length);
    this.setState({price: this.commonArray[nr]});
  }
  else if(op == "\"Rare\""){
    nr = Math.floor(Math.random()*this.rareArray.length);
    this.setState({price: this.rareArray[nr]});
  }
  else if(op == "\"Shiny\""){
    nr = Math.floor(Math.random()*this.shinyArray.length);
    this.setState({price: this.shinyArray[nr]});
  }
  else if(op == "\"Custom\""){
    nr = Math.floor(Math.random()*this.customArray.length);
    this.setState({price: this.customArray[nr]});
  }
  else{
    this.setState({price: 'ERROR'});
  }
}

onClick= (option) => {
  this.setState({click: true});
  this.rndSelection(option);
}

refresh = (navigation) =>{
  this.setState({click: false});
  navigation.navigate('Second');
}

render(){
  const {navigation, route} = this.props;
  const {text, option} = route.params;
  return (
    <View style={firstStyles.container}>
      <Text style={{ color: 'white', fontSize: 20,}}>Raffle a price!</Text>
      <Button
      title="Select Ticket"
```

```
        type="outline"
        buttonStyle={{backgroundColor: "#808080"}}
        titleStyle={{color: 'white'}}
        onPress={() => this.refresh(navigation)}
        />
        <Card containerStyle={{ backgroundColor: '#0099ff', borderRadius: 30,
width: 350, height: 250, alignItems: "center", justifyContent: "center",}}>
            <Text style={{color: 'white', fontSize: 30, alignItems: "cen-
ter"}}>{this.textSet(text)}</Text>
          </Card>
          <Button
          title="Raffle!"
          type="clear"
          titleStyle={{fontSize: 50, color: 'white',}}
          buttonStyle={{backgroundColor: '#00ccff', borderColor: '#00ccff',
alignItems: "center", justifyContent: "center", width: widthd, height: 200}}
          onPress={()=> this.onClick(option)}
          />
        <StatusBar style="auto" />
      </View>
    );

  }
}


//Second Page
function SecondScreen ({navigation}){
    return(
      <View style={secondStyles.container}>
        <TouchableOpacity
        style={{ padding: 20, borderWidth: 2, borderRadius: 10, borderColor:
'#cccccc', width: widthd-4, height: 80, justifyContent: 'flex-start'}}
        onPress={()=> navigation.navigate('Main', {text: 'Ticket Selected!', op-
tion: 'Cheap'})}
        >
          <Text style={{color: '#00ccff', fontSize: 20, textAlign:
'left'}}>Cheap</Text>
          <Text style={{ color: '#cccccc', fontSize: 15, textAlign:
'right',}}>2g</Text>
        </TouchableOpacity>
        <TouchableOpacity
        style={{ padding: 20, borderWidth: 2, borderRadius: 10, borderColor:
'#cccccc', width: widthd-4, height: 80, justifyContent: 'flex-start'}}
        onPress={()=> navigation.navigate('Main', {text: 'Ticket Selected!', op-
tion: 'Common'})}
        >
          <Text style={{color: '#00ccff', fontSize: 20, textAlign: 'left'}}>Com-
mon</Text>
          <Text style={{ color: '#cccccc', fontSize: 15, textAlign:
'right',}}>4g</Text>
        </TouchableOpacity>
        <TouchableOpacity
        style={{ padding: 20, borderWidth: 2, borderRadius: 10, borderColor:
'#cccccc', width: widthd-4, height: 80, justifyContent: 'flex-start'}}
        onPress={()=> navigation.navigate('Main', {text: 'Ticket Selected!', op-
tion: 'Rare'})}
        >
```

```
          <Text style={{color: '#00ccff', fontSize: 20, textAlign:
'left'}}>Rare</Text>
          <Text style={{ color: '#cccccc', fontSize: 15, textAlign:
'right',}}>6g</Text>
        </TouchableOpacity>
        <TouchableOpacity
        style={{ padding: 20, borderWidth: 2, borderRadius: 10, borderColor:
'#cccccc', width: widthd-4, height: 80, justifyContent: 'flex-start'}}
        onPress={()=> navigation.navigate('Main', {text: 'Ticket Selected!', op-
tion: 'Shiny'})}
        >
          <Text style={{color: '#00ccff', fontSize: 20, textAlign:
'left'}}>Shiny</Text>
          <Text style={{ color: '#cccccc', fontSize: 15, textAlign:
'right',}}>8g</Text>
        </TouchableOpacity>
        <TouchableOpacity
        style={{ padding: 20, borderWidth: 2, borderRadius: 10, borderColor:
'#cccccc', width: widthd-4, height: 80, justifyContent: 'flex-start'}}
        onPress={()=> navigation.navigate('Main', {text: 'Ticket Selected!', op-
tion: 'Custom'})}
        >
          <Text style={{color: '#00ccff', fontSize: 20, textAlign: 'left'}}>Cus-
tom</Text>
          <Text style={{ color: '#cccccc', fontSize: 15, textAlign:
'right',}}>10g</Text>
        </TouchableOpacity>
        <StatusBar style="auto" />
      </View>
      );
  }

//Settings Screen
function SettingsScreen(){
    return(
      <View style={secondStyles.container}>
        <Card
        containerStyle={{ padding: 20, borderWidth: 2, borderRadius: 10, border-
Color: '#cccccc', width: widthd-4, height: 60, justifyContent: 'flex-start'}}
        >
          <Text style={{color: '#0099ff', fontSize: 15, textAlign: 'left'}}>Shiny
Prices</Text>
          <Text style={{ color: '#00ccff', fontSize: 12, textAlign:
'right',}}>On</Text>
        </Card>
        <Card
        containerStyle={{ padding: 20, borderWidth: 2, borderRadius: 10, border-
Color: '#cccccc', width: widthd-4, height: 60, justifyContent: 'flex-start'}}
        >
          <Text style={{color: '#0099ff', fontSize: 15, textAlign: 'left'}}>Custom
Prices</Text>
          <Text style={{ color: '#00ccff', fontSize: 12, textAlign:
'right',}}>On</Text>
        </Card>
        <Card
        containerStyle={{ padding: 20, backgroundColor: "#ccf5ff", width: widthd,
height: 80, justifyContent: 'flex-start'}}
        >
```

```
          <Text style={{color: '#ffffff', fontSize: 20, textAlign: 'left', font-
Weight: 'bold',}}>Account</Text>
        </Card>
        <Card
        containerStyle={{ padding: 20, borderWidth: 2, borderRadius: 10, border-
Color: '#cccccc', width: widthd-4, height: 60, justifyContent: 'flex-start'}}
        >
          <Text style={{color: '#0099ff', fontSize: 15, textAlign: 'left'}}>Ac-
count</Text>
          <Text style={{ color: '#00ccff', fontSize: 12, textAlign: 'right',}}>ex-
ample@mail.com</Text>
        </Card>
        <Card
        containerStyle={{ padding: 20, borderWidth: 2, borderRadius: 10, border-
Color: '#cccccc', width: widthd-4, height: 60, justifyContent: 'flex-start'}}
        >
          <Text style={{color: '#0099ff', fontSize: 15, textAlign: 'left'}}>Noti-
fications</Text>
          <Text style={{ color: '#00ccff', fontSize: 12, textAlign:
'right',}}>Off</Text>
        </Card>
        <Card
        containerStyle={{ padding: 20, borderWidth: 2, borderRadius: 10, border-
Color: '#cccccc', width: widthd-4, height: 60, justifyContent: 'flex-start'}}
        >
          <Text style={{color: '#0099ff', fontSize: 15, textAlign:
'left'}}>Sound</Text>
          <Text style={{ color: '#00ccff', fontSize: 12, textAlign:
'right',}}>Off</Text>
        </Card>
        <Card
        containerStyle={{ padding: 20, borderWidth: 2, borderRadius: 10, border-
Color: '#cccccc', width: widthd-4, height: 60, justifyContent: 'flex-start'}}
        >
          <Text style={{color: '#0099ff', fontSize: 15, textAlign: 'left'}}>His-
tory</Text>
          <Text style={{ color: '#00ccff', fontSize: 12, textAlign:
'right',}}>Off</Text>
        </Card>
        <Card
        containerStyle={{ padding: 20, borderWidth: 2, borderRadius: 10, border-
Color: '#cccccc', width: widthd-4, height: 60, justifyContent: 'flex-start'}}
        >
          <Text style={{color: '#0099ff', fontSize: 15, textAlign:
'left'}}>Funds</Text>
          <Text style={{ color: '#00ccff', fontSize: 12, textAlign:
'right',}}>50g</Text>
        </Card>



      </View>

    );
  }

//App class
export default function App(){
    return(
```

```
     <NavigationContainer>
      <Stack.Navigator initialRouteName= "Main" >
        <Stack.Screen name="Main" component={MainScreen} initialParams={{text: 'No
Ticket Selected!', option: 'None', click: false,}} options={({navigation}) => ({
          title: "RAFFLER APP",
          headerStyle: {
            backgroundColor: '#0099ff',
          },
          headerTitleStyle: {
            fontWeight: 'bold',
            fontSize: 40,
            color: 'white',
            alignSelf: 'center',
          },
          headerRight: () => (
            <Icon
            name="settings"
            color="#cccccc"
            onPress={()=> navigation.navigate("Settings")}
            />
          )
        })} />
        <Stack.Screen name="Second" component={SecondScreen} options={({naviga-
tion}) => ({
          title: "SELECT TICKET",
          headerStyle: {
            backgroundColor: 'white',
          },
          headerTitleStyle: {
            fontWeight: 'bold',
            fontSize: 30,
            color: '#00ccff',
            alignSelf: 'center',
          },
          headerRight: () => (
            <Icon
            name="settings"
            color="#cccccc"
            onPress={()=> navigation.navigate("Settings")}
            />
          )
        })}/>
        <Stack.Screen name="Settings" component={SettingsScreen} options={{
          title: "SETTINGS",
          headerStyle: {
            backgroundColor: 'white',
          },
          headerTitleStyle: {
            fontWeight: 'bold',
            fontSize: 30,
            color: '#00ccff',
            alignSelf: 'center',
          },
        }}
        />
      </Stack.Navigator>
    </NavigationContainer>
    );
    }
```

```
const firstStyles = StyleSheet.create({
  container: {
    flex: 1,
    paddingTop: 30,
    backgroundColor: '#0099ff',
    alignItems: 'center',
    justifyContent: 'space-between',
  },
});

const secondStyles = StyleSheet.create({
  container: {
    flex: 1,
    paddingTop: 30,
    backgroundColor: 'white',
    alignItems: 'center',
    justifyContent: 'space-between',
  },
});
```

Raffler Flutter: Main

```
// Copyright 2018 The Flutter team. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.
import 'service_locator.dart';
import 'package:flutter/material.dart';
import 'dart:core';
import 'text_updater.dart';
import 'dart:math';

// class to launch app on
void main() {
  setupLocator();
  runApp(ParentState());
}

//homepage creator
class MyHome extends StatefulWidget {
  MyHome({Key key, @required this.ticketText, @required this.raffleList}) : su-
per(key: key);
  final String ticketText;
  final List<String> raffleList;

  HomePage createState() => HomePage();
}

//homepage code
class HomePage extends State<MyHome>{
  @override

  //price box text handler
  String text = "holder";
 String  _textCheck(){
    if (text == "holder") {
    text = widget.ticketText;
    return text;
    }
    else
      return text;
  }

  //random price picker
  randomize(){
    var rng = new Random();
    setState(() {
      text = widget.raffleList[new Random().nextInt(widget.raffleList.length)];
    });
  }

  // Main page layout
  //// Appbar layout and menu button
  //// Title text section
  //// Ticket selection button
  //// Price box
  //// Price raffle button
  Widget build(BuildContext context){
```

```
    return Scaffold(
       backgroundColor: Colors.blue,
       appBar: AppBar(
         backgroundColor: Colors.blue,
           title: Text(
             'Raffle App',
             style: TextStyle(
               fontWeight: FontWeight.bold,
               color: Colors.white,
               fontSize: 40,
             ),
           ),
         centerTitle: true,
         leading: new GestureDetector(
             onTap: () {locator<NavigationService>().navigateTo('menu');},
             child: Icon(
               Icons.menu,
               color: Colors.white,
             )
         ),
       ),
       body: Column(
         children: [
           titleSection,
         Container(
           padding: const EdgeInsets.only(top: 40, left: 40, right: 40, bottom:
100),
           child: Column(
             crossAxisAlignment: CrossAxisAlignment.center,
             children: [
               RaisedButton(
                 color: Colors.grey,
                 onPressed: () { locator<NavigationService>().navi-
gateTo('second');},
                 padding: EdgeInsets.all(20),
                 child: Text(
                   'Select Ticket',
                   style: TextStyle(
                     color: Colors.black,
                     fontSize: 15,
                     fontWeight: FontWeight.bold,
                   ),),),],),),
         Container(
           alignment: FractionalOffset.center,
           height: 150,
           width: 300,
           decoration: BoxDecoration(
             color: Colors.lightBlueAccent,
             border: Border.all(
               color: Colors.white,
               width: 2,
             ),
             borderRadius: BorderRadius.circular(12),
           ),
          child: Text(
             _textCheck(),
             textAlign: TextAlign.center,
             style: TextStyle(
                 color: Colors.white,
```

```
                fontSize: 40
            ),
          ),
        ),
          Expanded(
            child:  Align(
              alignment: FractionalOffset.bottomCenter,
              child: Column(
                crossAxisAlignment: CrossAxisAlignment.stretch,
                mainAxisSize: MainAxisSize.max,
                mainAxisAlignment: MainAxisAlignment.end,
                children: <Widget>[
                  RaisedButton(
                    color: Colors.lightBlueAccent,
                    onPressed: () => randomize(),
                    padding: EdgeInsets.all(40),
                    child: Text(
                      'RAFFLE!',
                      style: TextStyle(
                        color: Colors.white,
                        fontSize: 20,
                        fontWeight: FontWeight.bold,
                      ),),),],),),),
        ],),);
  }
}

//app text layout container
Widget titleSection = Container(
  padding: const EdgeInsets.all(32),
  child: Row(
    children: [
      Expanded(
        /*1*/
        child: Column(
          crossAxisAlignment: CrossAxisAlignment.center,
          children: [
            Text(
              'Raffle a price',
              style: TextStyle(
                color: Colors.white,
                fontSize: 20,
              ),),],),),
      /*3*/

    ],),);
```

Raffler Flutter: Menu page

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

class MenuPage extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Menu',
      home: Scaffold(
        appBar: AppBar(
          title: Text(
              'Settings',
            style: TextStyle(
              fontWeight: FontWeight.bold,
              color: Colors.lightBlueAccent,
            ),
          ),
          centerTitle: true,
          backgroundColor: Colors.white,
        ),
        body: Center(
          child:Column(
            children: <Widget>[
              Card(
                child: Column(
                  mainAxisSize: MainAxisSize.max,
                  children: <Widget>[
                    const ListTile(
                      title: Text(
                        'Shiny prices',
                        textAlign: TextAlign.left,
                        style: TextStyle(
                          color: Colors.blue,
                          fontWeight: FontWeight.bold,
                        ),
                      ),
                      subtitle: Text(
                        'On',
                        textAlign: TextAlign.right,
                        style: TextStyle(
                          color: Colors.lightBlueAccent,
                          fontStyle: FontStyle.italic,
                      ),),),],),),
              Card(
                child: Column(
                  mainAxisSize: MainAxisSize.max,
                  children: <Widget>[
                    const ListTile(
                      title: Text(
                        'Custom prices',
                        textAlign: TextAlign.left,
                        style: TextStyle(
                          color: Colors.blue,
                          fontWeight: FontWeight.bold,
                        ),
                      ),
```

```
        subtitle: Text(
          'On',
          textAlign: TextAlign.right,
          style: TextStyle(
            color: Colors.lightBlueAccent,
            fontStyle: FontStyle.italic,
          ),),),],),),
Card(
  color: Colors.blue[100],
  child: Column(
    mainAxisSize: MainAxisSize.max,
    children: <Widget>[
      Container(
        padding: EdgeInsets.all(20),
        child:
        const ListTile(
          title: Text(
            'Account',
            textAlign: TextAlign.left,
            style: TextStyle(
              color: Colors.white,
              fontWeight: FontWeight.bold,
              fontSize: 30,
            ),),),),],),),
Card(
  child: Column(
    mainAxisSize: MainAxisSize.max,
    children: <Widget>[
      const ListTile(
        title: Text(
          'Account',
          textAlign: TextAlign.left,
          style: TextStyle(
            color: Colors.blue,
            fontWeight: FontWeight.bold,
          ),
        ),
        subtitle: Text(
          'example@mail.com',
          textAlign: TextAlign.right,
          style: TextStyle(
            color: Colors.lightBlueAccent,
            fontStyle: FontStyle.italic,
          ),),),],),),
Card(
  child: Column(
    mainAxisSize: MainAxisSize.max,
    children: <Widget>[
      const ListTile(
        title: Text(
          'Notifications',
          textAlign: TextAlign.left,
          style: TextStyle(
            color: Colors.blue,
            fontWeight: FontWeight.bold,
          ),
        ),
        subtitle: Text(
          'Off',
```

```
          textAlign: TextAlign.right,
          style: TextStyle(
            color: Colors.lightBlueAccent,
            fontStyle: FontStyle.italic,
          ),),),],),),
Card(
  child: Column(
    mainAxisSize: MainAxisSize.max,
    children: <Widget>[
      const ListTile(
        title: Text(
          'Sound',
          textAlign: TextAlign.left,
          style: TextStyle(
            color: Colors.blue,
            fontWeight: FontWeight.bold,
          ),
        ),
        subtitle: Text(
          'Off',
          textAlign: TextAlign.right,
          style: TextStyle(
            color: Colors.lightBlueAccent,
            fontStyle: FontStyle.italic,
          ),),),],),),
Card(
  child: Column(
    mainAxisSize: MainAxisSize.max,
    children: <Widget>[
      const ListTile(
        title: Text(
          'History',
          textAlign: TextAlign.left,
          style: TextStyle(
            color: Colors.blue,
            fontWeight: FontWeight.bold,
          ),
        ),
        subtitle: Text(
          'Off',
          textAlign: TextAlign.right,
          style: TextStyle(
            color: Colors.lightBlueAccent,
            fontStyle: FontStyle.italic,
          ),),),],),),
Card(
  child: Column(
    mainAxisSize: MainAxisSize.max,
    children: <Widget>[
      const ListTile(
        title: Text(
          'Funds',
          textAlign: TextAlign.left,
          style: TextStyle(
            color: Colors.blue,
            fontWeight: FontWeight.bold,
          ),
        ),
        subtitle: Text(
```

```
                        '50g',
                        textAlign: TextAlign.right,
                        style: TextStyle(
                          color: Colors.lightBlueAccent,
                          fontStyle: FontStyle.italic,
                        ),),),],),),],),),),);
  }
}
```

Raffler Flutter: Second page

```
import 'package:flutter/material.dart';
import 'service_locator.dart';
import 'dart:core';

//second page builder
class MySecond extends StatefulWidget {

  MySecond({Key key, @required this.onChange, @required this.listSelect,}) :su-
per(key: key);
  final String onChange;
  Null Function(String s) listSelect;

  SecondPage createState() => SecondPage();
}

//app second page code
class SecondPage extends State<MySecond> {

  @override

  //call out function for selecting list
  _handlefunction(String s){
    widget.listSelect(s);
  }

  //price box text call out function
  _handletext(){
    widget.onChange;
  }

  // Second Page Layout
  //// Appbar and menu navigation
  //// Cheap ticket button
  //// Common ticket button
  //// Rare ticket button
  //// Shiny ticket button
  //// Custom ticket button
  Widget build(BuildContext context) {
    return MaterialApp(
      home: Scaffold(
        appBar: AppBar(
          title: Text(
              'Select Ticket',
            style: TextStyle(
              color: Colors.lightBlueAccent,
          ),
          ),
          centerTitle: true,
          backgroundColor: Colors.white,
          leading: new GestureDetector(
            onTap: () {locator<NavigationService>().navigateTo('menu');},
            child: Icon(
              Icons.menu,
              color: Colors.lightBlueAccent,
            )
```

```
  ),
),
body: Column(
  children:[
    Expanded(
      child: Align(
        alignment: FractionalOffset.bottomCenter,
        child: Column(
          crossAxisAlignment: CrossAxisAlignment.stretch,
          mainAxisSize: MainAxisSize.max,
          mainAxisAlignment: MainAxisAlignment.end,
          children: <Widget>[
            RaisedButton(
              color: Colors.white,
              onPressed: () {
                _handletext();
                _handlefunction("Cheap");
                locator<NavigationService>().navigateTo('');
              },
              padding: EdgeInsets.all(40),
              child: new Row(
                mainAxisAlignment: MainAxisAlignment.spaceBetween,
                children: <Widget>[
                  new Text(
                    'Cheap',
                    style: TextStyle(
                      color: Colors.lightBlueAccent,
                      fontSize: 20,
                      fontWeight: FontWeight.bold,
                    ),
                  ),
                  new Text(
                    '2g',
                    style: TextStyle(
                      color: Colors.grey,
                      fontSize: 15,
                      fontStyle: FontStyle.italic,
                    ),),],),),],),),),),
    Expanded(
      child: Align(
        alignment: FractionalOffset.bottomCenter,
        child: Column(
          crossAxisAlignment: CrossAxisAlignment.stretch,
          mainAxisSize: MainAxisSize.max,
          mainAxisAlignment: MainAxisAlignment.end,
          children: <Widget>[
            RaisedButton(
              color: Colors.white,
              onPressed: () {
                _handletext();
                _handlefunction("Common");
                locator<NavigationService>().navigateTo('');
              },
              padding: EdgeInsets.all(40),
              child: new Row(
                mainAxisAlignment: MainAxisAlignment.spaceBetween,
                children: <Widget>[
                  new Text(
                    'Common',
```

```
                  style: TextStyle(
                    color: Colors.lightBlueAccent,
                    fontSize: 20,
                    fontWeight: FontWeight.bold,
                  ),
                ),
                new Text(
                  '4g',
                  style: TextStyle(
                    color: Colors.grey,
                    fontSize: 15,
                    fontStyle: FontStyle.italic,
                  ),),],),),],),),),),
Expanded(
  child: Align(
    alignment: FractionalOffset.bottomCenter,
    child: Column(
      crossAxisAlignment: CrossAxisAlignment.stretch,
      mainAxisSize: MainAxisSize.max,
      mainAxisAlignment: MainAxisAlignment.end,
      children: <Widget>[
        RaisedButton(
          color: Colors.white,
          onPressed: () {
            _handletext();
            _handlefunction("Rare");
            locator<NavigationService>().navigateTo('');
          },
          padding: EdgeInsets.all(40),
          child: new Row(
            mainAxisAlignment: MainAxisAlignment.spaceBetween,
            children: <Widget>[
              new Text(
                'Rare',
                style: TextStyle(
                  color: Colors.lightBlueAccent,
                  fontSize: 20,
                  fontWeight: FontWeight.bold,
                ),
              ),
              new Text(
                '6g',
                style: TextStyle(
                  color: Colors.grey,
                  fontSize: 15,
                  fontStyle: FontStyle.italic,
                ),),],),),],),),),),
Expanded(
  child: Align(
    alignment: FractionalOffset.bottomCenter,
    child: Column(
      crossAxisAlignment: CrossAxisAlignment.stretch,
      mainAxisSize: MainAxisSize.max,
      mainAxisAlignment: MainAxisAlignment.end,
      children: <Widget>[
        RaisedButton(
          color: Colors.white,
          onPressed: () {
            _handletext();
```

```
                    _handlefunction("Shiny");
                    locator<NavigationService>().navigateTo('');
                  },
                  padding: EdgeInsets.all(40),
                  child: new Row(
                    mainAxisAlignment: MainAxisAlignment.spaceBetween,
                    children: <Widget>[
                      new Text(
                        'Shiny',
                        style: TextStyle(
                          color: Colors.lightBlueAccent,
                          fontSize: 20,
                          fontWeight: FontWeight.bold,
                        ),
                      ),
                      new Text(
                        '8g',
                        style: TextStyle(
                          color: Colors.grey,
                          fontSize: 15,
                          fontStyle: FontStyle.italic,
                        ),),],),),],),),),),
            Expanded(
              child: Align(
                alignment: FractionalOffset.bottomCenter,
                child: Column(
                  crossAxisAlignment: CrossAxisAlignment.stretch,
                  mainAxisSize: MainAxisSize.max,
                  mainAxisAlignment: MainAxisAlignment.end,
                  children: <Widget>[
                    RaisedButton(
                      color: Colors.white,
                      onPressed: () {
                        _handletext();
                        _handlefunction("Custom");
                        locator<NavigationService>().navigateTo('');
                      },
                      padding: EdgeInsets.all(40),
                      child: new Row(
                        mainAxisAlignment: MainAxisAlignment.spaceBetween,
                        children: <Widget>[
                          new Text(
                            'Custom',
                            style: TextStyle(
                              color: Colors.lightBlueAccent,
                              fontSize: 20,
                              fontWeight: FontWeight.bold,
                            ),
                          ),
                          new Text(
                            '10g',
                            style: TextStyle(
                              color: Colors.grey,
                              fontSize: 15,
                              fontStyle: FontStyle.italic,
                            ),),],),),],),),),),]),),);
  }
}
```

Raffler Flutter: Service locator

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

GetIt locator = GetIt();

class NavigationService {
  final GlobalKey<NavigatorState> navigatorKey = new GlobalKey<NavigatorState>();
  Future<dynamic> navigateTo(String routeName) {
    return navigatorKey.currentState.pushNamed(routeName);
  }
}

void setupLocator() {
  locator.registerLazySingleton(() => NavigationService());
}
```

Raffler Flutter: Text updater

```
import 'package:flutter/material.dart';
import 'main.dart';
import 'secondpage.dart';
import 'service_locator.dart';
import 'menupage.dart';

//parent state of all pages
class ParentState extends StatefulWidget{
  @override
  MyAppState createState()=> MyAppState();
}

//parent state code and navigation
class MyAppState extends State<ParentState>{

  //Price box text to indicate ticket selection
  String _ticketText = "Ticket not selected!";
  changeText(){
      _ticketText = "Ticket Selected!";
  }


  //Lists of price options
  List<String> pricesCheap = [
    'Red Fox',
    'Cross Fox',
    'Silver Fox',
    'Pearl Fox',
    'Naali Winter',
  ];

  List<String> pricesCommon = [
    'Burgundy Fox',
    'Amber Fox',
    'Shadow Fox',
    'Ice Fox',
    'Naali Summer',
  ];

  List<String> pricesRare = [
    'Platinum Fox',
    'Black Cross Fox',
    'Gold Fox',
    'Platinum Cross Fox',
    'Naali Autumn',
  ];

  List<String> pricesShiny = [
    'Red Marble Fox',
    'Marble Fox',
    'Burgundy Marble Fox',
    'Cross Marble Fox',
    'Naali Spring',
  ];
```

```
  List<String> pricesCustom = [
    'Custom Fox',
    'Custom Fox',
    'Custom Fox',
    'Naali Custom',
    'Naali Custom',
  ];

  //List to hold the price list selected, and indicator of no ticket selected
  List<String> holderList = ["Nothing Selected", "Nothing Selected", "Nothing Se-
lected", "Nothing Selected", "Nothing Selected",];

  //function for applying selected price list to the holder
  void applyList(String text){
    switch(text){
      case "Cheap": holderList = pricesCheap; break;
      case "Common": holderList = pricesCommon; break;
      case "Rare": holderList = pricesRare; break;
      case "Shiny": holderList = pricesShiny; break;
      case "Custom":  holderList = pricesCustom; break;
      default: holderList = [text];
    }
  }

  @override

  //App navigation and assigned functions
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Flutter layout demo',
      navigatorKey: locator<NavigationService>().navigatorKey,
      onGenerateRoute: (routeSettings){
        switch(routeSettings.name){
          case 'second':
            return MaterialPageRoute (builder: (context) => MySecond(onChange:
changeText(), listSelect: (String s) {applyList(s);},));
          case 'menu':
            return MaterialPageRoute (builder: (context) => MenuPage());
          default:
            return MaterialPageRoute(builder: (context)=> MyHome(ticketText:
_ticketText, raffleList: holderList));
        }
      },
      home: MyHome(ticketText: _ticketText, raffleList: holderList,),
    );
  }
}
```