

Client code

Ready-made client code — the platform controller, page messages, the coin balance, the reward button, the tutorial offer page and the purchase window — for the web, React, Cocos Creator and Unity.

These samples are the client side of the [guides](#). Most come as **TypeScript (web)**, **React** and **C# (Unity)** tabs, and there's a complete [Cocos Creator](#) version at the end. The in-game web view (the bottom sheet) has [its own page](#).

Platform controller

The heart of the client: it asks your server for the player's token, builds every page link, opens pages, and runs the first-launch choice on the splash ([Login](#)).

TypeScript (web)

```
// baziness.ts - the client controller.
const TOKEN_KEY = "mygame_baziness_token";
const CHOICE_KEY = "mygame_platform_choice";

export class BazinessPlatform {
  enabled = false;
  isLinked = false;
  private token: string | null = null;
  private webBase = "";
  private apiBase = "";
  private packageName = "";
  private loginGate: ((ok: boolean) => void) | null = null;
  private loginSucceeded = false;
  onChange: (() => void) | null = null;

  constructor(private sheet: BazinessSheet, private deviceId: string) {
    sheet.onMessage = (m) => this.handleMessage(m);
    sheet.onClosed = () => this.handleClosed();
  }

  get choiceMade() { return localStorage.getItem(CHOICE_KEY) === "1"; }
  markChoiceMade() { localStorage.setItem(CHOICE_KEY, "1"); }

  async loadConfig(): Promise<boolean> {
    try {
      const cfg = await fetch("/api/platform/config").then((r) => r.json());
      this.enabled = cfg.enabled;
      if (cfg.enabled) this.apply(cfg);
    } catch { this.enabled = false; }
    return this.enabled;
  }
}
```

```

    /** Resolve the token. Returns false when the page is about to reload into another account.
    */
    async init(): Promise<boolean> {
        // The Baziness app opens games with ?linkedToken=... - forward it once, then it is
        stripped on reload.
        const linkedToken = new URLSearchParams(location.search).get("linkedToken") ||
        localStorage.getItem(TOKEN_KEY);
        const resp = await fetch("/api/platform/token", {
            method: "POST", headers: { "Content-Type": "application/json" },
            body: JSON.stringify({ deviceId: this.deviceId, linkedToken }),
        }).then((r) => r.json()).catch(() => null);
        if (!resp?.enabled) { this.enabled = false; return true; }
        this.enabled = true;
        this.apply(resp);
        if (resp.switchToDeviceId && resp.switchToDeviceId !== this.deviceId) {
            this.switchAccount(resp.switchToDeviceId);
            return false;
        }
        if (resp.token) {
            this.token = resp.token;
            this.isLinked = !!resp.isLinkedToAccount;
            localStorage.setItem(TOKEN_KEY, resp.token);
        }
        this.onChange?.();
        return true;
    }

    /** Splash "Sign in with Baziness": resolves true once a login/link succeeded, false if
    closed. */
    async openLoginAndWait(): Promise<boolean> {
        if (!this.token) await this.init(); // the anonymous user's token becomes linkedToken
        if (!this.enabled || !this.webBase) return false;
        return new Promise((resolve) => {
            this.loginGate = resolve;
            this.loginSucceeded = false;
            this.sheet.open(this.pageUrl("/", { allowGuest: false }));
        });
    }

    /** Settings → "Connect your account": the profile page. */
    openAccount() { this.sheet.open(this.pageUrl("/profile", { redirect: "profile", allowGuest:
    false })); }

    pageUrl(path: string, o: { redirect?: string; allowGuest?: boolean; extra?: Record<string,
    string> } = {}) {
        const u = new URL(this.webBase + path);
        for (const [k, v] of Object.entries(o.extra ?? {})) u.searchParams.set(k, v);
        u.searchParams.set("packageName", this.packageName);
        u.searchParams.set("sdkType", "js");
        if (this.token) u.searchParams.set("linkedToken", this.token);
        if (o.redirect) u.searchParams.set("redirect", o.redirect);
        if (o.allowGuest === false) u.searchParams.set("allowGuest", "false");
        return u.toString();
    }

    private apply(c: { webBase?: string; apiBase?: string; packageName?: string }) {
        this.webBase = c.webBase ?? ""; this.apiBase = c.apiBase ?? ""; this.packageName =
        c.packageName ?? "";
    }

```

```
// handleMessage / handleClose / switchAccount: see "Messages from the Baziness page"
below.
}
```

React

```
// Splash.tsx – the first-time choice.
export function Splash({ onReady }: { onReady: (me: Me) => void }) {
  const [needsChoice, setNeedsChoice] = useState(false);
  const choice = useRef<() => void | null>(null);
  const loggedIn = useRef(false);

  useEffect(() => {
    (async () => {
      let me = (await authenticate()).me; // 1. your own account first
      if (await baziness.loadConfig()) { // 2. platform on?
        if (!baziness.choiceMade && !me.user.isBazinessLinked) {
          setNeedsChoice(true); // 3. ask once
          await new Promise<void>((r) => (choice.current = r));
          setNeedsChoice(false);
        }
        if (!(await baziness.init())) return; // 4. reloading into the linked
account
        if (loggedIn.current || baziness.isLinked !== me.user.isBazinessLinked) {
          me = await api<Me>("GET", "/api/me"); // 5. show the linked account
        }
      }
      onReady(me);
    })();
  }, [onReady]);

  const chooseGuest = () => { baziness.markChoiceMade(); choice.current?.(); };
  const chooseLogin = async () => {
    await baziness.init();
    if (!(await baziness.openLoginAndWait())) return; // closed without logging in → keep
the buttons
    loggedIn.current = true;
    choice.current?.();
  };

  return needsChoice ? (
    <div className="splash-choice">
      <button className="btn btn-green" onClick={chooseLogin}>ورود با حساب بازینس</button>
      <button className="btn btn-ghost" onClick={chooseGuest}>ادامه به عنوان مهمان</button>
    </div>
  ) : <ProgressBar />;
}
```

C# (Unity)

```
// BazinessLogin.cs – the same protocol from a Unity client. Your SERVER still does every
// platform call; Unity only talks to your server and shows the page in a WebView.
using System;
using System.Collections;
using UnityEngine;
using UnityEngine.Networking;

public class BazinessLogin : MonoBehaviour
{

```

```

const string TokenKey = "baziiness_token", ChoiceKey = "baziiness_choice";
[SerializeField] string serverUrl = "https://api.yourgame.com";
[SerializeField] BaziinessSheet sheet; // see the Bottom sheet page

public bool Enabled { get; private set; }
public bool IsLinked { get; private set; }
public string Token { get; private set; }
public string PackageName => packageName;
public string GameOfferUrl { get; private set; } // from /api/platform/token
string webBase, apiBase, packageName;
Action<bool> loginGate;

[Serializable] class TokenResponse {
    public bool enabled; public string webBase, apiBase, packageName, token,
switchToDeviceId;
    public long baziinessUserId; public bool isLinkedToAccount;
}

public IEnumerator Init(string deviceId)
{
    var body = $"{{\"deviceId\":\"{deviceId}\",\"linkedToken\":
{Quote(PlayerPrefs.GetString(TokenKey, null))}}}}";
    using var req = new UnityWebRequest($"{serverUrl}/api/platform/token", "POST");
    req.uploadHandler = new UploadHandlerRaw(System.Text.Encoding.UTF8.GetBytes(body));
    req.downloadHandler = new DownloadHandlerBuffer();
    req.SetRequestHeader("Content-Type", "application/json");
    yield return req.SendWebRequest();
    if (req.result != UnityWebRequest.Result.Success) { Enabled = false; yield break; }

    var r = JsonUtility.FromJson<TokenResponse>(req.downloadHandler.text);
    Enabled = r.enabled;
    if (!r.enabled) yield break;
    (webBase, apiBase, packageName) = (r.webBase, r.apiBase, r.packageName);
    if (!string.IsNullOrEmpty(r.switchToDeviceId) && r.switchToDeviceId != deviceId)
    {
        DeviceIdentity.Set(r.switchToDeviceId); // adopt the other account ...
        PlayerPrefs.DeleteKey(TokenKey);
        GameBoot.Restart(); // ... and boot again as that player
        yield break;
    }
    if (!string.IsNullOrEmpty(r.token)) { Token = r.token; IsLinked =
r.isLinkedToAccount; PlayerPrefs.SetString(TokenKey, r.token); }
}

public void OpenLogin(Action<bool> done)
{
    loginGate = done;
    sheet.Open(PageUrl("/", allowGuest: false), OnRawPageMessage, onClosed: () => {
loginGate?.Invoke(IsLinked); loginGate = null; });
}

public string PageUrl(string path, string redirect = null, bool allowGuest = true, string
extra = null)
{
    var url = $"{webBase}{path}?{extra} {(extra != null ? "&" : "")}packageName=
{Uri.EscapeDataString(packageName)}&sdkType=js";
    if (!string.IsNullOrEmpty(Token)) url += "&linkedToken=
{Uri.EscapeDataString(Token)}";
    if (redirect != null) url += "&redirect={redirect}";
    if (!allowGuest) url += "&allowGuest=false";
}

```

```

        return url;
    }

    public void OnRawPageMessage(string json) { /* see "Messages from the Baziness page"
below */ }
    static string Quote(string s) => string.IsNullOrEmpty(s) ? "null" : $"\"{s}\"";
}

```

Handling page messages

`close`, `login`, `token` and `logout`, as listed in [Baziness pages → messages](#).

TypeScript (web)

```

// Inside BazinessPlatform (continued).
const extractToken = (data: unknown): string | undefined =>
    typeof data === "string" ? data
    : data && typeof (data as { token?: unknown }).token === "string" ? (data as { token:
string }).token
    : undefined;

handleMessage(msg: { type: string; data?: unknown }) {
    const token = extractToken(msg.data);
    switch (msg.type) {
        case "close":
            this.sheet.close();
            break;
        case "login":
            if (token) void this.applyNewLogin(token); // signed into an account → switch or link
            else void this.linkedInPlace(); // linked onto the current player
            break;
        case "token":
            break; // silent handshake – NOT a login
        case "logout":
            this.sheet.close();
            localStorage.clear(); // identity, token, choice, tutorial
            flags...
            this.reloadClean(); // next boot = brand-new visitor
            break;
    }
}

private async applyNewLogin(token: string) {
    const result = await fetch("/api/platform/login", {
        method: "POST", headers: { "Content-Type": "application/json" },
        body: JSON.stringify({ deviceId: this.deviceId, token }),
    }).then((r) => r.json());
    if (result.switchToDeviceId && result.switchToDeviceId !== this.deviceId) {
        this.markChoiceMade();
        this.switchAccount(result.switchToDeviceId); // reloads
        return;
    }
    if (result.ok) {
        localStorage.setItem(TOKEN_KEY, token);
        await this.init(); // fresh token + linked flag
    }
}

```

```

    this.loginSucceeded = true;
    this.markChoiceMade();
    this.sheet.close(); // handleClosed resolves the login gate
  }
  // on failure keep the page open so the player can retry
}

private async linkedInPlace() {
  this.isLinked = true;
  this.loginSucceeded = true;
  this.markChoiceMade();
  if (this.loginGate) { await this.init(); this.sheet.close(); }
  this.onChange?.();
}

private handleClosed() {
  const gate = this.loginGate;
  this.loginGate = null;
  gate?.(this.loginSucceeded); // true only if a login actually happened
  this.onChange?.(); // coin balance / link state may have
  changed
}

private switchAccount(deviceId: string) {
  setDeviceId(deviceId); // your own identity store
  localStorage.removeItem(TOKEN_KEY);
  this.reloadClean();
}

private reloadClean() {
  const url = new URL(location.href);
  url.searchParams.delete("linkedToken"); // never re-apply a stale token
  url.searchParams.delete("device"); // a ?device= test override would undo the
  switch
  location.replace(url.toString());
}

```

React

```

// BazinessSheet.tsx – listening for page messages (only from the page's own origin).
function parse(data: unknown): { type: string; data?: unknown } | null {
  let value = data;
  if (typeof data === "string") {
    try { value = JSON.parse(data); } catch { return null; }
  }
  return value && typeof value === "object" && typeof (value as { type?: unknown }).type ===
  "string"
    ? (value as { type: string; data?: unknown })
    : null;
}

useEffect(() => {
  if (!url) return;
  const origin = new URL(url).origin;
  const onMessage = (event: MessageEvent) => {
    if (event.origin !== origin) return; // ignore HMR, extensions, other frames
    const msg = parse(event.data);
    if (msg) baziness.handleMessage(msg);
  };

```

```

window.addEventListener("message", onMessage);
return () => window.removeEventListener("message", onMessage);
}, [url]);

```

C# (Unity)

```

// Called by BazinessSheet with the raw JSON the page posted (see the Bottom sheet page).
// Uses Newtonsoft.Json (com.unity.nuget.newtonsoft-json) because `data` can be a string OR
an object.
using Newtonsoft.Json.Linq;

public void OnRawPageMessage(string json)
{
    JObject msg;
    try { msg = JObject.Parse(json); } catch { return; }
    var type = (string)msg["type"];
    var data = msg["data"];
    string token = data?.Type == JTokenType.String ? (string)data : (string)data?["token"];

    switch (type)
    {
        case "close": sheet.Close(); break;
        case "login":
            if (!string.IsNullOrEmpty(token)) StartCoroutine(ApplyNewLogin(token)); // POST
//api/platform/login
            else { IsLinked = true; StartCoroutine(Init(DeviceIdentity.Get()));
sheet.Close(); }
            break;
        case "token": break; // silent handshake - NOT a login
        case "logout":
            sheet.Close();
            PlayerPrefs.DeleteAll(); // forget identity, token and choice
            GameBoot.Restart(); // boot again as a new visitor
            break;
    }
}

```

Showing the balance

TypeScript (web)

```

// Inside BazinessPlatform: the player's Gita coin balance, straight from the platform. null
= unknown.
async getCoin(): Promise<number | null> {
    if (!this.enabled || !this.token || !this.apiBase) return null;
    try {
        const res = await fetch(`${this.apiBase}/user/get-coin`, {
            headers: { Authorization: `Bearer ${this.token}`, "Content-Type": "application/json" },
        });
        if (!res.ok) return null;
        const json = (await res.json()) as { amount?: number };
        return typeof json.amount === "number" ? json.amount : null;
    } catch {
        return null;
    }
}

```

```

    }
  }

  // Wiring: refresh on every relevant moment.
  baziness.onChange = () => void baziness.getCoin().then(renderCoinChip); // sheet closed,
  login, link
  window.addEventListener("focus", () => void baziness.getCoin().then(renderCoinChip));

```

React

```

// session.tsx - keep the external balance in context and refresh it on the right events.
const [gita, setGita] = useState<number | null>(null);

const refreshGita = useCallback(async () => {
  if (baziness.enabled) setGita(await baziness.getCoin());
}, []);

useEffect(() => {
  baziness.onChange = () => void refreshGita(); // fires when the sheet closes / login
  changes
  void refreshGita();
  const onFocus = () => void refreshGita();
  window.addEventListener("focus", onFocus);
  return () => window.removeEventListener("focus", onFocus);
}, [refreshGita]);

// Anywhere in the UI:
<span className="gita-chip">
  <CurrencyIcon currency="GITA_COIN" />
  <bdi>{gita === null ? "..." : faNum(gita)}</bdi>
</span>

```

C# (Unity)

```

using System;
using System.Collections;
using UnityEngine;
using UnityEngine.Networking;

[Serializable] class CoinResponse { public long amount; }

// apiBase and the user token come from /api/platform/token (see the Login page).
public IEnumerator GetCoin(Action<long?> done)
{
  if (!Enabled || string.IsNullOrEmpty(Token)) { done(null); yield break; }
  using var req = UnityWebRequest.Get($"{apiBase}/user/get-coin");
  req.SetRequestHeader("Authorization", $"Bearer {Token}");
  req.SetRequestHeader("Content-Type", "application/json");
  yield return req.SendWebRequest();
  if (req.result != UnityWebRequest.Result.Success) { done(null); yield break; }
  done(JsonUtility.FromJson<CoinResponse>(req.downloadHandler.text).amount);
}

// Refresh when the app regains focus and when the Baziness sheet closes.
void OnApplicationFocus(bool focus) { if (focus) StartCoroutine(GetCoin(UpdateCoinChip)); }

```

The reward button

TypeScript (web)

```
// Inside BazinessPlatform. GAME_NAME comes from /api/platform/config (or a constant).
openRewardSelection(): void {
  if (!this.enabled || !this.webBase) return;
  this.sheet.open(
    this.pageUrl("/shop", { redirect: "shop", extra: { game_name: this.gameName } }),
    BAZINESS_BOTTOM_BAR_CROP_PX, // optional: hide the platform's own tab bar (see below)
  );
}

// The lobby banner
rewardBanner.onclick = () => {
  if (baziness.enabled) baziness.openRewardSelection();
  else toast("انتخاب جایزه بعد از اتصال به بازی‌نس فعال می‌شود");
};
```

React

```
// Home.tsx – the «انتخاب جایزه» banner with the live balance.
const { gita } = useSession(); // from get-coin, see the Gita coins page

const openRewards = () => {
  if (baziness.enabled) baziness.openRewardSelection();
  else toast("انتخاب جایزه بعد از اتصال به بازی‌نس فعال می‌شود", "info");
};

return (
  <button className="hero-banner reward-banner" onClick={openRewards}>
    <span className="hero-text">
      <h2>انتخاب جایزه</h2>
      <span className="hero-sub">گینا کوین‌هایت را با جایزه‌ی واقعی عوض کن</span>
      <span className="gita-chip">
        <CurrencyIcon currency="GITA_COIN" size={30} />
        <bdi className="num">{gita === null ? "..." : faNum(gita)}</bdi>
      </span>
    </span>
    <img className="hero-art" src={art("ui/banner-rewards")} alt="" />
  </button>
);
```

C# (Unity)

```
// RewardBanner.cs – attach to the lobby button.
public class RewardBanner : MonoBehaviour
{
  [SerializeField] BazinessLogin platform;
  [SerializeField] BazinessSheet sheet;
  [SerializeField] TMPro.TMP_Text balanceLabel;
  [SerializeField] string gameName = "example";

  void OnEnable() => StartCoroutine(platform.GetCoin(ShowBalance));

  public void OnClick()
  {
  }
```

```

        if (!platform.Enabled) { Toast.Show("انتخاب جایزه بعد از اتصال به بازیس فعال میشود");
return; }
        var url = platform.PageUrl("/shop", redirect: "shop", extra: $"game_name=
{Uri.EscapeDataString(gameName)}");
        sheet.Open(url, platform.OnRawPageMessage, onClosed: () =>
StartCoroutine(platform.GetCoin(ShowBalance)));
    }

    void ShowBalance(long? amount) => balanceLabel.text = amount?.ToString("N0") ?? "...";
}

```

Hiding the bottom bar

The technique from [Reward page](#): make the frame taller than the view, and let the view clip it.

TypeScript (web)

```

/** Pixels cropped off the bottom of the embedded page. 0 = show the platform's tab bar. */
export const BAZINESS_BOTTOM_BAR_CROP_PX = 0;

// in BazinessSheet.open(url, cropBottomPx):
this.frame.style.marginBottom = cropBottomPx > 0 ? `${-cropBottomPx}px` : "";

```

Tutorial offer page

Open the offer page once during the tutorial, and give the starter pack when it closes ([Tutorial offer page](#)).

TypeScript (web)

```

// Inside BazinessPlatform.
gameOfferUrl = "";
private closeListeners = new Set<() => void>();

/** Run fn whenever the sheet closes; returns the unsubscribe function. */
onSheetClosed(fn: () => void) { this.closeListeners.add(fn); return () =>
this.closeListeners.delete(fn); }
// ...and in handleClosed(): for (const fn of [...this.closeListeners]) fn();

get canOpenSheet() { return this.enabled && !!this.webBase; }

/** Onboarding only. Adds the SDK parameters to the server-provided URL. */
openGameOffer() {
    if (!this.gameOfferUrl) return;
    const u = new URL(this.gameOfferUrl); // already has ?game_name=
    u.searchParams.set("packageName", this.packageName);
    u.searchParams.set("sdkType", "js"); // or its close button can't close
    the sheet
    if (this.token) u.searchParams.set("linkedToken", this.token);
    u.searchParams.set("allowGuest", "false");
    this.sheet.open(u.toString());
}

```

```

}

// Lobby: the coach-marked banner
function onRewardBannerTap() {
  if (onboarding.stage === "rewards") {
    onboarding.setStage("pack"); // persist first: a reload resumes at
    the pack
    if (baziness.canOpenSheet && baziness.gameOfferUrl) {
      const off = baziness.onSheetClosed(() => { off(); showStarterPack(); });
      baziness.openGameOffer();
    } else {
      showStarterPack(); // platform off: never block
    }
  }
  return;
}
baziness.openRewardSelection(); // after onboarding: the real shop
}

```

React

```

// Home.tsx: the onboarding branch of the «انتخاب جایزه» banner.
const openRewards = () => {
  if (coach === "rewards") {
    setOnboardingStage("pack"); // persisted in localStorage
    if (baziness.canOpenSheet && baziness.gameOfferUrl) {
      const off = baziness.onSheetClosed(() => {
        off();
        setCoach("pack"); // → StarterPackModal
      });
      setCoach("shop");
      baziness.openGameOffer();
    } else {
      setCoach("pack");
    }
    return;
  }
  if (baziness.enabled) baziness.openRewardSelection();
  else toast("انتخاب جایزه بعد از اتصال به بازیکنس فعال میشود", "info");
};

{coach === "rewards" && (
  <div className="coach-veil" aria-hidden="true">
    <div className="coach-bubble">
      <img src={asset("/assets/ui/tutorial-guide.png")} alt="" />
      <p>!بزن تا ببینی یا گیتا کوین‌هایت چه جایزه‌هایی می‌توانی بگیری </b></p>
    </div>
  </div>
)}

```

C# (Unity)

```

public void OnRewardBannerTap()
{
  if (Onboarding.Stage == OnboardingStage.Rewards)
  {
    Onboarding.SetStage(OnboardingStage.Pack); // persist before opening
    if (platform.Enabled && !string.IsNullOrEmpty(platform.GameOfferUrl))
    {

```

```

        var url = platform.GameOfferUrl
            + "&packageName={Uri.EscapeDataString(platform.PackageName)}&sdkType=js"
            + "&linkedToken={Uri.EscapeDataString(platform.Token ??
        "" )}&allowGuest=false";
        sheet.Open(url, platform.OnRawPageMessage, onClosed: StarterPack.Show);
    }
    else StarterPack.Show(); // never block onboarding
    return;
}
rewardBanner.OnClick(); // after onboarding: /shop
}

```

Purchase window

Open the tab **inside the tap**, start the purchase on your server, send the tab to the payment page, and verify when the player returns ([Baziness Pay](#)).

TypeScript (web)

```

// purchase.ts - the purchase popup.
export function buy(productId: string, ui: PurchaseUi) {
    const tab = window.open("about:blank", "_blank"); // 1. synchronously, no
    "noopener"
    let paymentId: string | null = null;
    let started = false;
    let verifying = false;

    const verify = async (manual: boolean) => {
        if (!started || verifying || !paymentId) return;
        verifying = true;
        ui.showChecking();
        try {
            const r = await api("POST", "/api/shop/purchase/verify", { paymentId });
            if (r.status === "FULFILLED") { cleanup(); ui.success(r.granted); void
refreshWallets(); return; }
            if (manual) ui.toast("پرداختی پیدا نشد. اگر پرداخت کرده‌ای چند لحظه بعد دوباره بررسی کن.");
            ui.showWaiting();
        } catch (err) {
            if (manual) ui.error(err);
            ui.showWaiting();
        } finally { verifying = false; }
    };

    const onFocus = () => void verify(false);
    const onVisible = () => document.visibilityState === "visible" && void verify(false);
    const cleanup = () => { removeEventListener("focus", onFocus);
document.removeEventListener("visibilitychange", onVisible); };
    addEventListener("focus", onFocus);
    document.addEventListener("visibilitychange", onVisible);

    ui.showOpening(() => verify(true) /* «پرداخت کردم» */, () => { void verify(false); cleanup();
} /* close */);
    (async () => {

```

```

    try {
      const r = await api("POST", "/api/shop/purchase/start", { productId }); // 2.
      paymentId = r.paymentId;
      if (tab && !tab.closed) tab.location.href = r.url;
      else window.open(r.url, "_blank", "noopener");
      started = true;
      ui.showWaiting(); // 3.
    } catch (err) {
      tab?.close(); cleanup(); ui.error(err);
    }
  })();
}

```

React

```

// Shop.tsx + PurchaseModal.tsx: open the tab in the click, then mount the modal.
export function openPayTab(): Window | null {
  return window.open("about:blank", "_blank"); // must run inside the click handler
}

// <button onClick={() => setBuying({ product, payTab: openPayTab() })}>...</button>

export function PurchaseModal({ product, payTab, onClose }: Props) {
  const { setWallets, toast, refresh } = useSession();
  const [phase, setPhase] = useState<"opening" | "waiting" | "checking" | "done">("opening");
  const paymentId = useRef<string | null>(null);
  const started = useRef(false);
  const verifying = useRef(false);

  const verify = useCallback(async (manual: boolean) => {
    if (!paymentId.current || verifying.current || !started.current) return;
    verifying.current = true;
    setPhase("checking");
    try {
      const res = await api<VerifyPurchaseResponse>("POST", "/api/shop/purchase/verify", {
        paymentId: paymentId.current });
      setWallets(res.wallets);
      if (res.status === "FULFILLED") { setPhase("done"); toast("خرید با موفقیت انجام شد", "success"); void refresh(); return; }
      if (manual) toast("پرداختی پیدا نشد. اگر پرداخت کرده‌ای چند لحظه بعد دوباره بررسی کن", "info");
      setPhase("waiting");
    } catch (err) {
      if (manual) toast(errorMessage(err), "error");
      setPhase("waiting");
    } finally { verifying.current = false; }
  }, [refresh, setWallets, toast]);

  useEffect(() => { // start once
    (async () => {
      try {
        const res = await api<StartPurchaseResponse>("POST", "/api/shop/purchase/start", {
          productId: product.id });
        paymentId.current = res.paymentId;
        if (payTab && !payTab.closed) payTab.location.href = res.url;
        else window.open(res.url, "_blank", "noopener");
        started.current = true;
        setPhase("waiting");
      }
    })();
  }, [product.id, payTab]);
}

```

```

    } catch (err) { payTab?.close(); toast(errorMessage(err), "error"); onClose(); }
  })();
}, []);

useEffect(() => { // verify when the player comes back
  const onFocus = () => void verify(false);
  const onVisible = () => document.visibilityState === "visible" && void verify(false);
  window.addEventListener("focus", onFocus);
  document.addEventListener("visibilitychange", onVisible);
  return () => { window.removeEventListener("focus", onFocus);
document.removeEventListener("visibilitychange", onVisible); };
}, [verify]);

const close = () => { if (phase !== "done" && paymentId.current) void verify(false);
onClose(); };

return (
  <Modal open onClose={close}>
    <h3>{product.title}</h3>
    {phase === "done" ? <p>✓ به حسابت اضافه شد</p>
      : phase === "waiting" ? (<
        <p>صفحه‌ی پرداخت بازینس باز شد. بعد از پرداخت به بازی برگرد</p>
        <button onClick={() => void verify(true)}>پرداخت کردم</button>
      </>
      : <Spinner /> }
    </Modal>
  );
}

```

C# (Unity)

```

// PurchaseFlow.cs – on mobile the payment page opens in the system browser, so there is no
// popup blocker. Verify when the app regains focus, plus a manual «پرداخت کردم» button.
public class PurchaseFlow : MonoBehaviour
{
    [SerializeField] GameApi api; // your HTTP wrapper around /api/shop/purchase/*
    string paymentId;
    bool started, verifying;

    public IEnumerator Buy(string productId)
    {
        StartResponse r = null;
        yield return api.Post<StartResponse>("/api/shop/purchase/start", new { productId }, x
=> r = x);
        if (r == null) { Toast.Show("خطا در اتصال به درگاه"); yield break; }
        paymentId = r.paymentId;
        Application.OpenURL(r.url); // {base}/buy?sku&uid&pname, built by YOUR server
        started = true;
        PurchasePanel.ShowWaiting(onPaid: () => StartCoroutine(Verify(manual: true)));
    }

    void OnApplicationFocus(bool focus)
    {
        if (focus && started) StartCoroutine(Verify(manual: false)); // back from the
browser
    }

    IEnumerator Verify(bool manual)

```

```

{
    if (verifying || string.IsNullOrEmpty(paymentId)) yield break;
    verifying = true;
    VerifyResponse v = null;
    yield return api.Post<VerifyResponse>("/api/shop/purchase/verify", new { paymentId },
x => v = x);
    verifying = false;
    if (v?.status == "FULFILLED") { started = false; PurchasePanel.Success(v.granted);
Wallet.Set(v.wallets); }
    else if (manual) Toast.Show("پرداختی پیدا نشد. اگر پرداخت کرده‌ای چند لحظه بعد دوباره بررسی کن.");
}

[System.Serializable] class StartResponse { public string paymentId, url; }
[System.Serializable] class VerifyResponse { public string status; public Grant[]
granted; public Wallets wallets; }
}

```

Cocos Creator

A complete set for Cocos Creator 3.x, in three files:

File	Role
<code>BazinessSDK.ts</code>	a persistent component: opens pages, routes their messages, reads the balance
<code>BazinessWebView.ts</code>	the component on your WebView prefab: loading overlay, close button, slide animations, messages
<code>BazinessController.ts</code>	boot wiring: <code>linkedToken</code> from the address, <code>/api/platform/token</code> , sign-in, sign-out, account switch

The prefab needs a `WebView`, a loading overlay node, a close button, and an `Animation` with `Open` and `Close` clips (slide up / slide down). On web builds the WebView is an `iframe`, so page messages arrive as `window message` events.

BazinessSDK.ts

```

// BazinessSDK.ts - a persistent component (put it on a node in the first scene).
// Opens Baziness pages in a WebView prefab, routes their messages and reads the Gita coin balance.
import { _decorator, Component, director, find, instantiate, Prefab, EventTarget } from "cc";
import { BazinessWebView } from "../BazinessWebView";
const { ccclass, property } = _decorator;

export const bazinessEvents = new EventTarget(); // "login" | "logout" | "closed"

```

```

const WEB = { dev: "https://platform-web-dev.karizmastudios.org", prod:
"https://baziness.com" };
const API = { dev: "https://platform-dev.karizmastudios.org/api", prod:
"https://platform.karizmastudio.com/api" };

@ccclass("BazinessSDK")
export class BazinessSDK extends Component {
  @property(Prefab) webViewPrefab: Prefab = null!;

  static instance: BazinessSDK;
  static packageName = "com.yourstudio.yourgame";
  static devMode = true;
  static token: string | null = null; // the player's user token, from YOUR server
  static isLinked = false;

  private view: BazinessWebView | null = null;

  onLoad() {
    BazinessSDK.instance = this;
    director.addPersistRootNode(this.node);
  }

  /** Call after your server answered /api/platform/token. */
  static initialize(token: string, isLinked: boolean) {
    this.token = token;
    this.isLinked = isLinked;
  }

  static openLogin() {
    this.instance.open(`${this.web()}`, { allowGuest: false });
  }
  static openProfile() {
    this.instance.open(`${this.web()}/profile`, { redirect: "profile", allowGuest: false });
  }
  static openRewards(gameName: string) {
    this.instance.open(`${this.web()}/shop`, { redirect: "shop", extra: { game_name: gameName
} });
  }

  /** The Gita coin balance, straight from the platform with the player's own token. */
  static async getCoin(): Promise<number | null> {
    if (!this.token) return null;
    try {
      const res = await fetch(`${this.devMode ? API.dev : API.prod}/user/get-coin`, {
        headers: { Authorization: `Bearer ${this.token}`, "Content-Type": "application/json"
},
      });
      return res.ok ? ((await res.json()).amount ?? null) : null;
    } catch {
      return null;
    }
  }

  private static web() {
    return this.devMode ? WEB.dev : WEB.prod;
  }

  private open(base: string, o: { redirect?: string; allowGuest?: boolean; extra?:
Record<string, string> } = {}) {
    if (this.view) return; // one page at a time

```

```

const q = new URLSearchParams(o.extra);
q.set("packageName", BazinessSDK.packageName);
q.set("sdkType", "js"); // the page posts close / login / logout / token
if (BazinessSDK.token) q.set("linkedToken", BazinessSDK.token);
if (o.redirect) q.set("redirect", o.redirect);
if (o.allowGuest === false) q.set("allowGuest", "false");

const node = instantiate(this.webViewPrefab);
node.setParent(find("Canvas"));
this.view = node.getComponent(BazinessWebView!);
this.view.onMessage = (m) => this.onMessage(m);
this.view.onClosed = () => {
  this.view = null;
  bazinessEvents.emit("closed");
};
this.view.open(`${base}?${q}`);
}

private onMessage(msg: { type: string; data?: unknown }) {
  const token = typeof msg.data === "string" ? msg.data : (msg.data as { token?: string } | undefined)?.token;
  switch (msg.type) {
    case "close":
      this.view?.close();
      break;
    case "login":
      if (token) {
        this.view?.close(true);
        bazinessEvents.emit("login", token); // → POST /api/platform/login on your server
      } else {
        BazinessSDK.isLinked = true; // linked in place
      }
      break;
    case "logout":
      this.view?.close(true);
      bazinessEvents.emit("logout");
      break;
    case "token":
      break; // the page's silent handshake – not a login
  }
}
}
}

```

BazinessWebView.ts

```

// BazinessWebView.ts – the component on the WebView prefab (WebView + loading overlay + close button).
import { _decorator, Animation, Button, Component, Node, WebView } from "cc";
const { ccclass, property } = _decorator;

@ccclass("BazinessWebView")
export class BazinessWebView extends Component {
  @property(WebView) webView: WebView = null!;
  @property(Node) loadingOverlay: Node = null!;
  @property(Button) closeButton: Button = null!;
  @property(Animation) animation: Animation = null!; // "Open" slides up, "Close" slides down

  onMessage: ((m: { type: string; data?: unknown }) => void) | null = null;
  onClosed: (() => void) | null = null;
}

```

```

private origin = "";
private closeTimer: ReturnType<typeof setTimeout> | null = null;
private listener = (e: MessageEvent) => this.onWindowMessage(e);

open(url: string) {
  this.origin = new URL(url).origin;
  this.webView.node.active = false;
  this.closeButton.node.active = false;
  // Our own close button appears after a few seconds: a way out even if the page never
  loads.
  this.closeTimer = setTimeout(() => {
    if (this.closeButton?.isValid) this.closeButton.node.active = true;
  }, 3000);
  this.closeButton.node.on(Button.EventType.CLICK, () => this.close(), this);
  this.webView.node.on(WebView.EventType.LOADED, () => {
    this.loadingOverlay.active = false;
    this.webView.node.active = true;
  }, this);
  this.webView.node.on(WebView.EventType.ERROR, () => this.close(), this);
  window.addEventListener("message", this.listener); // on web builds the WebView is an
  iframe
  this.webView.url = url;
  this.animation.play("Open");
}

close(immediately = false) {
  window.removeEventListener("message", this.listener);
  if (this.closeTimer) clearTimeout(this.closeTimer);
  if (!immediately) this.animation.play("Close");
  setTimeout(() => {
    this.onClosed?.();
    if (this.node?.isValid) this.node.destroy();
  }, immediately ? 0 : 500);
}

private onWindowMessage(e: MessageEvent) {
  if (e.origin !== this.origin) return; // only the page we opened
  try {
    const msg = typeof e.data === "string" ? JSON.parse(e.data) : e.data;
    if (msg && typeof msg.type === "string") this.onMessage?.(msg);
  } catch {
    /* not a Baziness message */
  }
}
}

```

BazinessController.ts

```

// BazinessController.ts – boot-time wiring between your server, the SDK and the page
messages.
import { BazinessSDK, bazinessEvents } from "../BazinessSDK";
import { GameApi } from "../GameApi"; // your HTTP client for /api/*
import { LocalData } from "../LocalData"; // your localStorage wrapper

export class BazinessController {
  /** Returns true when it started an account switch or a reset – stop your own boot then. */
  static async initialize(): Promise<boolean> {
    // The Baziness app opens games with ?linkedToken=... . If it doesn't match the token this

```

```

device
    // already holds, a different Baziness account is arriving: start clean before going on.
    const fromUrl = new URLSearchParams(location.search).get("LinkedToken");
    const saved = LocalData.get("baziness_token");
    if (fromUrl && saved && fromUrl !== saved) {
        this.resetAndReload();
        return true;
    }

    const resp = await GameApi.post("/api/platform/token", { LinkedToken: fromUrl ?? saved
});
    if (!resp.enabled) return false; // platform off: no Baziness UI
    if (resp.switchToDeviceId) {
        this.switchTo(resp.switchToDeviceId);
        return true;
    }
    if (resp.token) {
        LocalData.set("baziness_token", resp.token);
        BazinessSDK.initialize(resp.token, resp.isLinkedToAccount);
    }

    bazinessEvents.on("login", async (token: string) => {
        const r = await GameApi.post("/api/platform/login", { token }); // the server validates
it
        if (r.switchToDeviceId) this.switchTo(r.switchToDeviceId);
        else if (r.ok) {
            LocalData.set("baziness_token", token);
            await this.initialize();
        }
    });
    bazinessEvents.on("logout", () => this.resetAndReload());
    return false;
}

private static switchTo(deviceId: string) {
    LocalData.clear();
    LocalData.set("device_id", deviceId);
    this.reload();
}

private static resetAndReload() {
    LocalData.clear(); // the next boot is a brand-new visitor
    this.reload();
}

private static reload() {
    const url = new URL(location.href);
    url.searchParams.delete("LinkedToken");
    location.replace(url.toString()); // or reload your loading scene
}
}

```