LVA.studio™ — MasterPlan & Developer Implementation Guide

Prepared for: LVA.studio Team — Daniel (LVAdev) Date: August 12, 2025

This document bundles the LVA Studio MasterPlan summary with detailed developer instructions, Velo (Wix) integration code, and a standalone iframe HTML/CSS/JS UI to support a client-facing Roadmap/Progress Tracker and an Admin editor page. Contents: - Executive summary - Roadmap & data model - Velo (Admin & Client) code snippets - iframe (HTML/CSS/JS) UI snippets - postMessage protocol and integration steps - Visuals (timeline illustration and legend) - Next steps and checklist

Executive Summary

LVA.studio's MasterPlan defines a client website development workflow and pricing model. Our technical project builds a Client Website Development Tracker that lets clients view their project's progress inside their Wix Member profile and lets internal admins update the roadmap from a protected Admin page. Key goals: - Keep all secure operations (auth, DB reads/writes, file uploads) inside Wix Velo. - Use a standalone HTML/CSS/JS iframe for flexible UI (timeline, animations). - Sync data between Velo and iframe via window.postMessage. - Store roadmap JSON in the 'Main' collection field: 'roadmapProgress' (JSON string or object).

Roadmap Milestones & Statuses

The project's roadmap steps and allowed statuses (canonical values) are: 1. Application Received — applicationreceived 2. Waiting — waiting 3. In Progress — inprogress 4. Testing — testing 5. Looking for Bugs — lookingforbugs 6. Under Maintenance — undermaintenance 7. Done — done Each roadmap step model: - title: string - status: enum (see above) - percent: integer (0-100)

Data Model — Example 'roadmapProgress' JSON

```
[
  { "title": "Application Received", "status": "applicationreceived", "percent": 100 },
  { "title": "Domain & Email Setup", "status": "done", "percent": 100 },
  { "title": "Version 1.0 Draft & Setup", "status": "inprogress", "percent": 40 },
  { "title": "Version 2.0 & 3.0 Feedback Loops", "status": "waiting", "percent": 0 },
  { "title": "Final Edits & SEO", "status": "waiting", "percent": 0 },
  { "title": "Launch + Google Campaign", "status": "waiting", "percent": 0 }
]
```

Admin Page (Velo) — UI Elements & IDs

Add these elements to a protected Admin page (Wix Editor): - Dropdown: #clientSelect — populate with clients from 'Main' (label: name, value: _id) - Repeater: #roadmapRepeater containing: - Text: #stepTitle - Dropdown: #statusDropdown - Slider: #percentSlider - Button: #saveBtn (saves roadmapProgress) - Optional: #createDefaultBtn (preload default roadmap template) - Ensure page is accessible only by site collaborators or authenticated admins.

Admin Page — Velo Code (snippet)

```
import wixData from 'wix-data';

let selectedClientId;
let roadmapSteps = [];

$w.onReady(() => {
    loadClients();

    $w('#clientSelect').onChange(() => {
        selectedClientId = $w('#clientSelect').value;
        loadRoadmap(selectedClientId);
    });

    $w('#saveBtn').onClick(() => {
        saveRoadmap();
    });
});

function loadClients() {
    wixData.query("Main")
        .ascending("name")
        .find()
        .then(res => {
            const options = res.items.map(client => {
                return { label: client.name, value: client._id };
            });
            $w('#clientSelect').options = options;
        });
}

function loadRoadmap(clientId) {
    wixData.get("Main", clientId)
        .then(client => {
            try {
                roadmapSteps = typeof client.roadmapProgress === 'string'
                    ? JSON.parse(client.roadmapProgress)
                    : client.roadmapProgress || [];
            } catch (e) {
                roadmapSteps = [];
            }

            $w('#roadmapRepeater').data = roadmapSteps;

            $w('#roadmapRepeater').onItemReady(($item, itemData, index) => {
                $item('#stepTitle').text = itemData.title;

                $item('#statusDropdown').options = [
                    { label: "Application Received", value: "applicationreceived" },
                    { label: "Waiting", value: "waiting" },
                    { label: "In Progress", value: "inprogress" },
                    { label: "Done", value: "done" },
                    { label: "Testing", value: "testing" },
                    { label: "Looking for bugs", value: "lookingforbugs" },
                    { label: "Under Maintenance", value: "undermaintenance" }
                ];
                $item('#statusDropdown').value = itemData.status;

                $item('#percentSlider').min = 0;
                $item('#percentSlider').max = 100;
                $item('#percentSlider').value = itemData.percent;

                $item('#statusDropdown').onChange((event) => {
                    roadmapSteps[index].status = event.target.value;
                });

                $item('#percentSlider').onChange((event) => {
                    roadmapSteps[index].percent = event.target.value;
                });
            });
        });
}

function saveRoadmap() {
    if (!selectedClientId) {
        console.warn("No client selected");
        return;
    }

    wixData.update("Main", {
        _id: selectedClientId,
        roadmapProgress: JSON.stringify(roadmapSteps)
    })
    .then(() => {
        console.log("Roadmap updated successfully");
    })
    .catch(err => {
        console.error("Failed to update roadmap", err);
    });
}
```

Client Page — Velo + iFrame Integration

Client page responsibilities (in Velo): - Authenticate member (getCurrentMemberId). - Fetch member item from Main collection. - Post message to iframe with 'memberData' or 'roadmap'. - Listen for messages from iframe to update data in Main. ID: set the HTML iframe element's ID to #profileIframe in the Wix editor.

```
import wixData from 'wix-data';
import { getCurrentMemberId } from 'public/repeatFunctions.js';

let currentMemberId;
let memberData = {};

$w.onReady(async function () {
    currentMemberId = await getCurrentMemberId();
    memberData = await wixData.get("Main", currentMemberId);

    $w('#profileIframe').onMessage((event) => {
        // iframe signals ready
        if (event.data === "ready") {
            $w('#profileIframe').postMessage({ type: 'memberData', payload: memberData });
        }

        // save request from iframe
        if (event.data && event.data.type === 'saveData') {
            const updated = { ...memberData, ...event.data.payload };
            wixData.update('Main', updated)
                .then(() => {
                    // optionally reply to iframe
                    $w('#profileIframe').postMessage({ type: 'saveComplete', payload: updated });
                });
        }
    });
});
```

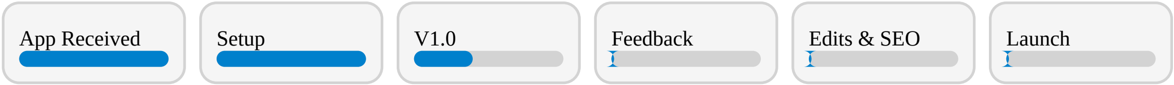
iframe — HTML/CSS/JS (standalone) Example

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title>Client Roadmap Editor</title>
<style>
  body{font-family:Arial,Helvetica,sans-serif;padding:14px;}
  .timeline{max-width:720px;margin:0 auto;}
  .step{border-radius:10px;padding:10px;margin:8px 0;background:#f4f6f8;border:1px solid #e1e6ea;}
  .progress{height:10px;background:#e7eef6;border-radius:6px;overflow:hidden;margin-top:8px;}
  .progress-fill{height:100%;width:0%;background:linear-gradient(90deg,#00bfff,#6cf0ff);transition:width .5s;}
  button{padding:8px 12px;border-radius:6px;border:none;background:#0b61a4;color:white;}
</style>
</head>
<body>
<h3>Project Roadmap</h3>
<div id="container" class="timeline"></div>
<button id="saveBtn">Save Changes</button>

<script>
let memberData = null;

// Tell parent we are ready
window.parent.postMessage("ready", "*");

window.addEventListener("message", (event) => {
  if (!event.data) return;
  if (event.data.type === 'memberData') {
    memberData = event.data.payload;
    renderRoadmap(memberData.roadmapProgress || []);
  }
});

function renderRoadmap(steps) {
  const container = document.getElementById('container');
  container.innerHTML = '';
  steps.forEach((s, i) => {
    const el = document.createElement('div');
    el.className = 'step';
    el.innerHTML = `
      <strong>${s.title}</strong>
      <div>Status: <select data-index="${i}" class="status">${s.status}</select></div>
      <div class="progress"><div class="progress-fill" style="width:${s.percent}%"></div></div>
      <div>Percent: <input data-index="${i}" class="percent" type="number" min="0" max="100" value="${s.percent}
    `;
    container.appendChild(el);
  });

  // populate status selects with canonical values
  document.querySelectorAll('.status').forEach(sel => {
    sel.innerHTML = [
      {l:'Application Received',v:'applicationreceived'},
      {l:'Waiting',v:'waiting'},
      {l:'In Progress',v:'inprogress'},
      {l:'Testing',v:'testing'},
      {l:'Looking for bugs',v:'lookingforbugs'},
      {l:'Under Maintenance',v:'undermaintenance'},
      {l:'Done',v:'done'}
    ].map(o=>`<option value="${o.v}">${o.l}</option>`).join('');
    sel.value = steps[sel.dataset.index].status;
  });
}

document.getElementById('saveBtn').addEventListener('click', () => {
  const container = document.getElementById('container');
  const updated = [];
  container.querySelectorAll('.step').forEach((stepEl, i) => {
    const title = stepEl.querySelector('strong').textContent;
    const status = stepEl.querySelector('.status').value;
    const percent = Number(stepEl.querySelector('.percent').value) || 0;
    updated.push({ title, status, percent });
  });

  // Post updated roadmap back to parent for Velo to save
  window.parent.postMessage({ type: 'saveData', payload: { roadmapProgress: JSON.stringify(updated) } }, '*');
  alert('Saved (sent to parent for DB update)');
});
</script>
</body>
</html>
```

postMessage Protocol (messages between Velo <-> iframe)

Standard message types and payloads: 1. From iframe -> parent (Velo) - "ready" : string (iframe signals it's loaded) - { type: 'saveData', payload: { roadmapProgress: '...JSON string...' } } : request parent to save data - { type: 'uploadImage', payload: { base64: '...', filename: 'avatar.png' } } : request parent to upload (optional) 2. From parent (Velo) -> iframe - { type: 'memberData', payload: { ...member record... } } : send member + roadmap data - { type: 'saveComplete', payload: { ...updated member... } } : reply that save completed successfully Security notes: - parent (Velo) must validate incoming messages origin and payload. - never allow iframe to write directly to DB; parent must perform all writes.

Visual — Sample Timeline (legend)



Legend: Application Received (green) — Waiting (grey) — In Progress (blue) — Testing (orange) — Looking for bugs (red) — Under Maintenance (purple) — Done (green).

Integration Steps — Quick Checklist

- Create 'roadmapProgress' field in Main collection (Text or JSON).
- Build Admin page with elements: #clientSelect, #roadmapRepeater, #saveBtn.
- Add Client profile page with an HTML iframe element id #profileIframe; upload iframe HTML to site files or external host.
- Implement Velo handlers: fetch member data, postMessage to iframe, listen for saveData and update Main.
- Secure admin page for site collaborators/admins only.
- Add default roadmap template loader for new clients (optional).
- Test message flows with console logs and sample clients before going live.

Next Steps & Optional Enhancements

- Embed image upload flow: iframe sends base64 to Velo; Velo saves to Media Manager and returns URL.
- Add email notifications when status changes (Velo backend job).
- Add scheduled 'pause' logic when client delays feedback (timeline freeze).
- Add audit log in Main collection to keep change history for roadmapProgress.
- Polish iframe UI: animations, SVG icons, accessibility (ARIA).

Appendix — Canonical Status List (CSV)

```
Application Received,applicationreceived
Waiting,waiting
In Progress,inprogress
Done,done
Testing,testing
Looking for bugs,lookingforbugs
Under Maintenance,undermaintenance
```