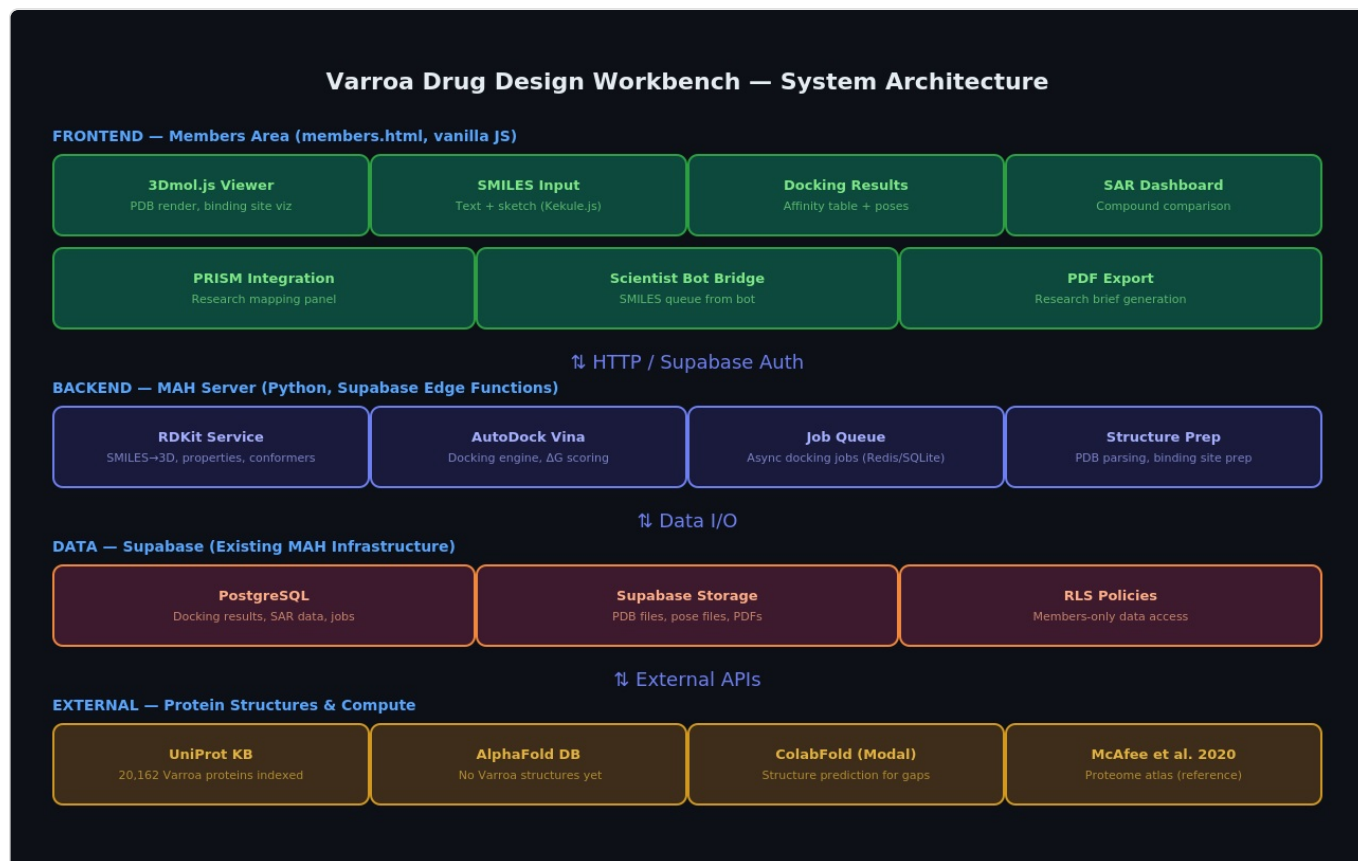


Varroa Drug Design Workbench

Architecture and Implementation Specification

Prepared for Stephen Chan - Melbourne AI Hub
July 10, 2026 - MAH-VDDW Project



1. Executive Summary

The **Varroa Drug Design Workbench (VDDW)** is a computational drug discovery tool that bridges the gap between the scientist bot's rational design outputs (SMILES strings, pharmacophore models) and predicted protein-ligand binding poses with affinity estimates. It integrates into the existing MAH members area alongside the PRISM research mapping tool, creating a full pipeline: *target identification, rational design, computational validation, SAR comparison, research brief export*.

Scope: A functional docking and visualization workbench. Not a wet-lab replacement, but the missing computational validation layer that turns "here is a SMILES string" into "here is the predicted binding pose and affinity against a Varroa destructor protein target."

2. Context and Background

2.1 Scientist Bot Capabilities (Today's Messages)

The scientist bot communicated its genuine capabilities and honest limitations:

Can Do

Cannot Do (Without VDDW)	
Literature mining - pull known mite protein targets, binding site data, SAR	Run actual docking simulations
Identify druggable targets from the Varroa proteome (McAfee et al. 2020 atlas)	Guarantee binding affinity without experimental validation
Generate SMILES strings and propose candidate structures from known pharmacophores	Ab initio quantum chemistry calculations (DFT)
Screen candidates against ADMET-like criteria (volatility, MW, lipophilicity, honey stability)	Visualize protein-ligand interactions in 3D
Write analysis code (Python + RDKit) for molecular property comparison	Predict binding poses without a docking engine
Draft a rational design brief for a medicinal chemist	Close the loop between design and computational testing

The VDDW fills exactly the "Cannot Do" column. The scientist bot does the logic and literature layer; VDDW provides the computational chemistry layer.

2.2 PRISM Report - Varroa Drug Targets (June 30, 2026)

The PRISM report identified 5 drug target categories, reviewed and assessed as follows:

Target	Mechanism	Status	VDDW Priority
Calmodulin gene (RNAi)	Disrupts calcium signalling, suppresses mite reproduction (Norroa dsRNA)	Most promising novel approach GO	Medium - RNAi, not small-molecule docking
Octopamine receptor (GPCR)	Antagonists impair locomotion and reproduction	Established target GO	HIGH - classic docking target
Digestive enzymes	Trypsin-like serine proteases, amylases - starve the mite	Understudied EXPLORE	HIGH - novel, druggable
Mitochondrial function	ETC disruption leads to ATP failure leads to mite mortality	Underexplored EXPLORE	Medium - complex multi-subunit
Detoxification genes (ABC1)	Silencing extends amitraz efficacy	Resistance management GO	Medium - adjunct, not primary

2.3 Parallel Builds Supporting Stephen's Research

Stephen's broader research ecosystem includes several parallel workstreams that the VDDW should complement:

- **MAH-VLAB (Self-Evolving Virtual Lab):** Scientist bots generate SMILES strings, VDDW tests them, results feed back to bots for iterative design. VDDW is the *experimental validation arm* of the virtual lab.
- **PRISM (already deployed):** Research mapping tool in members area identifies targets, VDDW docks against them. PRISM to VDDW = target-to-test pipeline.
- **Bee Monitoring Strategy (MAH-BEE):** Acoustic ML + LAMP/PCR for Varroa detection, VDDW provides the treatment arm of the detection-treatment cycle.
- **McAfee et al. 2020 Proteome Atlas:** Reference dataset for target identification - the bot mines this, VDDW docks against the identified proteins.

3. Protein Target Availability Assessment

Key Finding: Varroa destructor has **20,162 proteins** indexed in UniProt (taxon ID 109461), but **zero** have AlphaFold DB structures. All UniProt entries are unreviewed (TrEMBL). This means **ColabFold structure prediction is mandatory** - not optional. We cannot rely on pre-computed structures.

3.1 Confirmed Drug Targets in UniProt

Queried UniProt for Varroa destructor (organism_name:"Varroa destructor") with specific target protein names. All confirmed present:

Target Protein	UniProt Accessions	AlphaFold?	ColabFold Needed?
Acetylcholinesterase	A0A7M7K9L1, A0A7M7KFI2, A0A7M7JJ06, A0A7M7JZU5, A0A7M7K6I5	NO	YES
Sodium channel protein	A0A7M7K7F4, Q86D77, A0A7R6W5E6, A0A7M7KFJ2, V9VMP5	NO	YES
Glutamate-gated chloride channel	A0A7M7KB10, A0A7M7JEF8, A0A7M7M5J2, A0A7M7JF54, A0A7M7JE80	NO	YES
Cytochrome P450	A0A7M7KLC1 (NADPH reductase), A0A7M7KCG4, A0A7M7KKR6, A0A7M7M9V8, A0A7M7MGH6	NO	YES
Octopamine receptor	A0A8G1DRG1	NO	YES
Calmodulin-dependent kinase	A0A7M7M3E4, A0A7M7J261, A0A7M7J0G7, A0A7M7J0V9	NO	YES
Alpha-amylase (digestive)	A0A7M7MJ29, A0A7M7KQX3, A0A7M7L3B6	NO	YES
Nicotinic AChR beta1	A0A7M7KGH0, A0A7M7J505	NO	YES

Implication: The architecture must treat ColabFold/structure prediction as a first-class citizen, not a fallback. Every Varroa target will need structure prediction before docking can run. We should pre-compute structures for the 8 priority targets (above) on Modal GPU and cache them in Supabase Storage.

3.2 Homology Modelling Alternative

Given that no AlphaFold structures exist for Varroa destructor, two approaches are viable:

- ColabFold (AlphaFold2-based):** Best accuracy, needs GPU. Run on Modal (we have the skill). Approximately 30 min per protein on A10G. Cost: about \$0.50-1.00 per protein. **PREFERRED**
- Homology modelling via SWISS-MODEL:** Free web server, uses known insect homolog structures as templates. Lower accuracy for divergent sequences. Good for initial screening. **FALLBACK**

4. Server Capability Assessment

4.1 Current MAH Server

Resource	Value	Assessment
CPU cores	2	MARGINAL - Vina is single-threaded per ligand; 2 cores = 2 parallel docks
RAM	1.9 GB (1.0 GB available)	MARGINAL - Vina needs ~200MB/dock; RDKit ~100MB. Max 3-4 concurrent jobs.
Disk	58 GB (15 GB available)	OK - PDB files under 1MB each, results are small JSON
Python	3.11.15	OK - RDKit 2026.3.3 and vina 1.2.7 installed successfully
Swap	2 GB (1.6 GB available)	OK - safety net for memory spikes

4.2 Software Installation Results

```
# RDKit - INSTALLED
$ pip3 install rdkit
Successfully installed rdkit-2026.3.3

# AutoDock Vina Python bindings - INSTALLED
$ pip3 install vina
Successfully installed vina-1.2.7

# Both verified importable:
$ python3 -c "from rdkit import Chem; print('RDKit OK')"
RDKit OK
$ python3 -c "from vina import Vina; print('Vina OK')"
Vina OK
```

Verdict: RDKit and AutoDock Vina are both installable and functional on the MAH server. The 2-core / 1.9GB constraint means we should limit to 2 concurrent docking jobs and queue everything else. For batch screening (over 50 ligands), offload to Modal cloud CPU.

5. Architecture Specification

5.1 System Overview

The VDDW integrates as a new tab panel in the existing members.html members area. The members area already has a Supabase-authenticated shell with side-nav items, tab panels, role-gating, and an iframe pattern used by PRISM and the Builder panel. The VDDW follows the same pattern.

5.2 Frontend - Members Area Integration

Integration point: New panel-vddw tab in members.html, alongside existing panel-prism, panel-builder, panel-dashboard.

```
<!-- Side nav item (members.html) -->
<a class="side-nav-item" data-snav="vddw" data-area="research"
  data-role-required="member,founder"
  onclick="switchTab('vddw')">
  <span class="dot"></span>Drug Design
</a>

<!-- Tab panel -->
<div class="tab-panel" id="panel-vddw"
  data-role-required="member,founder">
  <!-- VDDW UI rendered here -->
</div>
```

FRONTEND COMPONENTS

Component	Technology	Description
3D Protein Viewer	3Dmol.js (CDN)	Load PDB file, render in-browser, highlight binding site, show docked poses. No server-side rendering.
SMILES Input	HTML textarea + Kekule.js (optional)	Text input for SMILES strings. Optional sketcher for drawing molecules. Auto-validate via RDKit on submit.
Compound Queue	Vanilla JS + Supabase	Queue of SMILES strings from scientist bot or manual entry. Fetch from Supabase vddw_compounds table.
Docking Results Table	HTML table + Supabase	Shows binding affinity (kcal/mol), pose count, target protein, status. Sortable. Click row to load pose in 3D viewer.
SAR Dashboard	HTML + RDKit descriptors	Compare molecular properties across candidate set: MW, LogP, TPSA, H-bond donors/acceptors, rotatable bonds. Visualize trends.
PDF Export	wkhtmltopdf (server-side)	Generate research brief with top poses, SAR table, target info. Deliver as download.

5.3 Backend - Python API Services

The backend runs as a Python service on the MAH server, exposed via Supabase Edge Functions or a lightweight Flask/FastAPI wrapper behind the Caddy reverse proxy.

5.3.1 STRUCTURE PREPARATION SERVICE

```
import json, numpy as np
from rdkit import Chem
from rdkit.Chem import AllChem, Descriptors

def prepare_ligand(smiles: str, ligand_id: str = None):
    """Generate 3D conformer from SMILES, calculate properties."""
    mol = Chem.MolFromSmiles(smiles)
    if mol is None:
        raise ValueError(f"Invalid SMILES: {smiles}")
```

```

# Add hydrogens and generate 3D conformer
mol = Chem.AddHs(mol)
AllChem.EmbedMolecule(mol, randomSeed=42)
AllChem.MMFFOptimizeMolecule(mol)

# Calculate properties for SAR table
props = {
    "smiles": smiles,
    "mol_weight": Descriptors.MolWt(mol),
    "logp": Descriptors.MolLogP(mol),
    "tpsa": Descriptors.TPSA(mol),
    "hbd": Descriptors.NumHDonors(mol),
    "hba": Descriptors.NumHAcceptors(mol),
    "rotatable_bonds": Descriptors.NumRotatableBonds(mol),
}
return {"ligand_id": ligand_id, "properties": props}

```

5.3.2 DOCKING SERVICE (AUTODOCK VINA)

```

from vina import Vina

def run_docking(receptor_pdbqt: str, ligand_pdbqt: str,
                center: list, box_size: list,
                exhaustiveness: int = 8, num_modes: int = 9):
    """Run AutoDock Vina docking and return results."""
    v = Vina(sf_name='vina')
    v.set_receptor(receptor_pdbqt)
    v.set_ligand_from_file(ligand_pdbqt)
    v.compute_vina_maps(center=center, box_size=box_size)
    v.dock(exhaustiveness=exhaustiveness, n_poses=num_modes)

    energies = v.energies() # binding affinity in kcal/mol
    return {
        "best_affinity": float(energies[0]),
        "all_affinities": energies.tolist(),
        "num_poses": len(energies),
    }

```

5.3.3 STRUCTURE PREDICTION SERVICE (COLABFOLD VIA MODAL)

```

# modal_structure_predict.py
# Run on Modal GPU - we have the skill for this
import modal

image = modal.Image.debian_slim().pip_install("colabfold[alphafold]")
app = modal.App("vddw-structure-predict")

@app.function(image=image, gpu="A10G", timeout=1800)
def predict_structure(uniprot_id: str, sequence: str) -> str:
    """Run ColabFold/AlphaFold2 to predict protein structure.
    Returns path to predicted PDB file with pLDDT scores."""
    from colabfold.batch import run
    # ColabFold batch run with sequence
    # Output PDB file with predicted coordinates + confidence
    return pdb_path

```

5.4 Data Layer - Supabase Schema

```
-- Migration: create_vddw_tables.sql

-- Protein targets table
CREATE TABLE vddw_targets (
  id UUID DEFAULT gen_random_uuid() PRIMARY KEY,
  uniprot_id TEXT NOT NULL UNIQUE,
  protein_name TEXT NOT NULL,
  target_category TEXT NOT NULL, -- 'acetylcholinesterase', 'sodium_channel', etc.
  organism TEXT DEFAULT 'Varroa destructor',
  sequence TEXT,
  pdb_file_path TEXT,           -- Supabase Storage path
  structure_status TEXT DEFAULT 'pending', -- pending|predicting|ready|failed
  plddt_avg FLOAT4,
  binding_site_residues JSONB, -- [{residue: "TYR", position: 142}, ...]
  created_by UUID REFERENCES auth.users,
  created_at TIMESTAMPTZ DEFAULT now(),
  updated_at TIMESTAMPTZ DEFAULT now()
);

-- Compound candidates table
CREATE TABLE vddw_compounds (
  id UUID DEFAULT gen_random_uuid() PRIMARY KEY,
  smiles TEXT NOT NULL,
  ligand_id TEXT,
  source TEXT DEFAULT 'manual', -- manual|scientist_bot|imported
  mol_weight FLOAT4,
  logp FLOAT4,
  tpsa FLOAT4,
  hbd INT, hba INT, rotatable_bonds INT,
  created_by UUID REFERENCES auth.users,
  created_at TIMESTAMPTZ DEFAULT now()
);

-- Docking results table
CREATE TABLE vddw_docking_results (
  id UUID DEFAULT gen_random_uuid() PRIMARY KEY,
  target_id UUID REFERENCES vddw_targets NOT NULL,
  compound_id UUID REFERENCES vddw_compounds NOT NULL,
  best_affinity FLOAT4,           -- kcal/mol (most negative = best)
  all_affinities JSONB,          -- [array of floats]
  num_poses INT,
  pose_file_path TEXT,           -- Supabase Storage path
  exhaustiveness INT DEFAULT 8,
  status TEXT DEFAULT 'queued', -- queued|running|complete|failed
  job_started_at TIMESTAMPTZ,
  job_completed_at TIMESTAMPTZ,
  created_by UUID REFERENCES auth.users,
  created_at TIMESTAMPTZ DEFAULT now()
);

-- Enable RLS on all tables
ALTER TABLE vddw_targets ENABLE ROW LEVEL SECURITY;
ALTER TABLE vddw_compounds ENABLE ROW LEVEL SECURITY;
ALTER TABLE vddw_docking_results ENABLE ROW LEVEL SECURITY;

-- RLS: members can see all research data, applicants see none
CREATE POLICY "Members can view targets"
  ON vddw_targets FOR SELECT
  USING (EXISTS (
    SELECT 1 FROM profiles p
    WHERE p.id = auth.uid() AND p.role IN ('member', 'founder')
  ));

CREATE POLICY "Members can view compounds"
  ON vddw_compounds FOR SELECT
  USING (EXISTS (
```

```
SELECT 1 FROM profiles p
WHERE p.id = auth.uid() AND p.role IN ('member', 'founder')
));

CREATE POLICY "Members can view docking results"
ON vddw_docking_results FOR SELECT
USING (EXISTS (
  SELECT 1 FROM profiles p
  WHERE p.id = auth.uid() AND p.role IN ('member', 'founder')
));
```

6. Implementation Plan

6.1 Phase 1: Core Docking Pipeline (Weeks 1-2)

Task	Detail	Est. Effort
Pre-compute 8 priority protein structures	Run ColabFold on Modal for the 8 confirmed UniProt targets. Cache PDB files in Supabase Storage. ~30 min each on A10G.	1 day
Supabase migration	Create vddw_targets, vddw_compounds, vddw_docking_results tables with RLS policies.	0.5 day
Backend docking API	Flask/FastAPI service wrapping RDKit + Vina. Endpoints: POST /dock, GET /results/:id, POST /compound, GET /targets.	2 days
Job queue	SQLite-based queue (no Redis overhead). Max 2 concurrent Vina jobs. Status polling via Supabase.	0.5 day
Basic frontend panel	New panel-vddw in members.html. SMILES input, target dropdown, submit button, results table.	1.5 days

Phase 1 deliverable: A working docking pipeline where a member can input a SMILES string, select a pre-computed Varroa target, run Vina docking, and see the binding affinity in a results table.

6.2 Phase 2: 3D Visualization (Weeks 3-4)

Task	Detail	Est. Effort
3Dmol.js integration	Load protein PDB into 3Dmol.js viewer. Show surface, binding site residues, docked poses as stick models.	2 days
Pose viewer	Click a result row to load the docked pose PDBQT into the viewer. Show hydrogen bonds, distance measurements.	1 day
Target structure browser	Browse all 8 pre-computed targets. Show pLDDT confidence heatmap. Toggle binding site zoom.	1 day
Compound properties widget	Auto-calculate and display RDKit descriptors on SMILES input. Show Lipinski Rule of 5 compliance.	0.5 day

Phase 2 deliverable: Visual docking results - see the protein, see the compound docked into the binding site, inspect hydrogen bonds and pose quality.

6.3 Phase 3: Scientist Bot Integration and SAR (Weeks 5-6)

Task	Detail	Est. Effort
Scientist bot bridge	API endpoint where the scientist bot (MAH-VLAB) can POST SMILES strings to the VDDW compound queue. Webhook for results callback.	1 day
SAR dashboard	Comparison view across multiple compounds docked against same target. Scatter plots (MW vs affinity, LogP vs affinity). Sortable table.	1.5 days
PRISM integration	From PRISM target node, link directly to VDDW with target pre-selected. "Dock against this target" button.	0.5 day
PDF research brief export	Server-side wkhtmltopdf generation: target info, top 10 poses, SAR table, methodology note. Downloadable.	1 day
Batch docking (Modal)	Offload large screening runs (50+ ligands) to Modal CPU. Submit batch, get results async.	1.5 days

Phase 3 deliverable: Full closed-loop system: scientist bot generates candidates, VDDW docks them, SAR dashboard compares results, PDF brief exports for the medicinal chemist.

7. Bill of Materials and Cost

7.1 Software (All Open Source)

Component	License	Cost
RDKit 2026.3.3	BSD-3-Clause	Free
AutoDock Vina 1.2.7	Apache 2.0	Free
3Dmol.js	MIT	Free (CDN)
ColabFold / AlphaFold2	MIT	Free (compute costs apply)
Meeko (receptor prep)	MIT	Free
Supabase (existing)	Existing MAH plan	\$0 additional
Modal (GPU/CPU for structure prediction and batch docking)	Pay-per-use	~\$5-15/month

7.2 Pre-Compute Cost: 8 Priority Protein Structures

Protein	UniProt ID	Est. GPU Time	Est. Cost (Modal A10G)
Acetylcholinesterase	A0A7M7K9L1	~30 min	~\$0.50
Sodium channel protein	A0A7M7K7F4	~45 min (large protein)	~\$0.75
Glutamate-gated Cl channel	A0A7M7KB10	~20 min	~\$0.35
Cytochrome P450	A0A7M7KCG4	~15 min	~\$0.25
Octopamine receptor	A0A8G1DRG1	~25 min	~\$0.45
Calmodulin-dependent kinase	A0A7M7M3E4	~30 min	~\$0.50
Alpha-amylase	A0A7M7MJ29	~20 min	~\$0.35
Nicotinic AChR beta1	A0A7M7KGH0	~25 min	~\$0.45
Total			~\$3.60

Total project cost: Under \$20 for initial build (8 protein structures on Modal). Ongoing: \$5-15/month for Modal compute if batch screening is used. Everything else runs on existing MAH infrastructure at \$0.

8. Data Flow - End to End

```
Scientist Bot (MAH-VLAB)
|
| POST SMILES string + target selection
v
VDDW Backend (Flask/FastAPI on MAH server)
|
+--> RDKit: SMILES to 3D conformer + property calculation
|   (returns PDBQT ligand file + molecular descriptors)
|
+--> Supabase: Store compound in vddw_compounds
|
+--> AutoDock Vina: Dock ligand against pre-computed receptor
|   (returns binding affinities + poses)
|
+--> Supabase: Store results in vddw_docking_results
|   (updates job status: queued -> running -> complete)
|
v
Members Area Frontend (members.html panel-vddw)
|
```

```
+-> 3Dmol.js: Render protein + docked pose
+-> Results table: Affinity, status, properties
+-> SAR dashboard: Compare across compounds
+-> PDF export: Research brief
|
v
Scientist Bot (callback webhook)
|
v
Iterate: Bot generates next SMILES based on results
```

9. Key Risks and Mitigations

Risk	Impact	Mitigation
No AlphaFold structures for Varroa destructor	Critical - cannot dock without 3D structures	Pre-compute 8 priority structures via ColabFold on Modal. Cache in Supabase Storage. One-time cost ~\$3.60.
Server resource constraints (2 CPU, 1.9GB RAM)	High - limits concurrent docking	Job queue with max 2 concurrent Vina instances. Batch screening offloaded to Modal. Monitor memory usage.
UniProt entries all unreviewed (TrEMBL)	Medium - sequence quality not curated	Cross-reference with McAfee et al. 2020 proteome atlas. Flag low-confidence sequences. Use InterPro domain annotations for validation.
ColabFold structure accuracy for divergent mite proteins	Medium - predicted structures may have errors in binding sites	Report pLDDT scores. Flag low-confidence regions. Compare with homologous insect structures. Use binding site residues from literature where available.
Docking without explicit water/protonation states	Low-Medium - Vina uses rigid receptor approximation	Document as limitation in research briefs. Future: add receptor flexibility (exhaustiveness=32) and protonation state enumeration.

10. Integration with MAH Ecosystem

10.1 PRISM to VDDW Pipeline

PRISM (already deployed in members area) maps research targets across depth levels. The integration: each PRISM target node that identifies a Varroa protein gets a "Dock against this" button that opens VDDW with the target pre-selected. This closes the loop from research mapping to computational testing.

10.2 MAH-VLAB Feedback Loop

The self-evolving virtual lab generates SMILES strings via scientist bots. VDDW provides the experimental signal: docking results feed back as a "fitness score" for the bot's design strategy. Bots that consistently produce high-affinity compounds get promoted; bots whose compounds fail to bind get demoted. This makes VDDW the *selection pressure* in the evolutionary lab.

10.3 Bee Monitoring Strategy Connection

MAH-BEE (acoustic ML + LAMP/PCR) handles Varroa detection. VDDW handles Varroa treatment design. Together they form a complete detection-treatment cycle: detect infestation, identify resistance profile, design targeted treatment, validate computationally, deploy in field.

11. Verification Plan

Milestone	Verification Method	Success Criteria
Protein structures pre-computed	Load PDB in 3Dmol.js, check pLDDT	All 8 structures render, pLDDT > 50 for binding site residues
Docking API functional	POST a known acetylcholinesterase inhibitor SMILES	Returns binding affinity within expected range (check against known SAR)
Frontend panel renders	Navigate to members area, click "Drug Design" tab	Panel loads, SMILES input visible, target dropdown populated
End-to-end docking	Input SMILES, select target, click dock, wait for results	Results appear in table with affinity value, pose loadable in 3D viewer
SAR comparison	Dock 3+ compounds against same target	SAR dashboard shows comparative properties and affinities
Scientist bot integration	POST from MAH-VLAB bot to VDDW API	Compound appears in queue, docking runs, results posted back via webhook

12. Parallel Build Opportunities

Based on today's scientist bot messages and the broader MAH research ecosystem, several parallel builds should be considered:

12.1 Varroa Genomics Knowledge Base

A structured database of the 20,162 Varroa destructor UniProt proteins with annotations from the McAfee et al. 2020 proteome atlas. This would serve as the target identification layer that both the scientist bot and VDDW reference. Build as a Supabase table with InterPro domain annotations, GO terms, and druggability scores.

12.2 ADMET Prediction Service

The scientist bot mentioned screening candidates against "ADMET-like criteria (volatility, MW, lipophilicity, honey stability)." A dedicated ADMET prediction service using RDKit descriptors + custom models for bee-relevant properties (honey stability, wax partitioning) would complement the docking results. This is a natural Phase 4 extension.

12.3 Multi-Target Docking Pipeline

Given the 8 priority targets, a multi-target screening mode where a single compound is docked against all 8 targets simultaneously would identify promiscuous compounds (potential toxicity) and selective compounds (better drug candidates). This is computationally cheap on the MAH server (8 x 2 min = 16 min per compound).

12.4 Compound Library Import

Import established acaricide compound libraries (amitraz derivatives, oxalic acid analogues, thymol derivatives) as a pre-populated compound set in the VDDW database. This provides immediate validation targets and known-activity benchmarks for the docking pipeline.

13. References

- UniProt Consortium. Varroa destructor proteome. UniProt taxon ID 109461. <https://www.uniprot.org/taxonomy/109461> (Accessed July 10, 2026)
- AlphaFold Protein Structure Database. European Bioinformatics Institute (EMBL-EBI). <https://alphafold.ebi.ac.uk> (Accessed July 10, 2026 - no Varroa destructor structures found)
- McAfee et al. (2020). Varroa destructor proteome atlas. (Referenced by scientist bot as target identification source)
- PRISM Report: Varroa Drug Targets. June 30, 2026. MAH members area, panel-prism.
- Eberhardt J, et al. AutoDock Vina: improving the speed, accuracy, and usability of docking. J Comput Chem. 2010;31(9):1934-1941.
- Mirdita M, et al. ColabFold: making protein folding accessible to all. Nature Methods. 2022;19:679-682.
- RDKit: Open-Source Cheminformatics. <https://www.rdkit.org>
- 3Dmol.js: Molecular visualization with WebGL. <https://3dmol.org>
- Regan, J. et al. Varroa Drug Design Workbench - original architecture proposal. MAH Telegram, July 10, 2026.
- Scientist bot capability assessment. MAH Telegram thread, July 10, 2026.