# BOTFORCE Unity

Consolidated Specification + Sprint Backlog
(Designed for implementation via Claude Code)

Version: v1.0
Date: 27 January 2026
Owner: BOTFORCE GmbH (Vienna, Austria)

## Table of Contents

- Tip: In Word, you can insert an automatic Table of Contents using the Heading styles in this document.

## 1. Executive Summary

BOTFORCE Unity is a production-ready invoicing, time tracking, expense management, and accounting-prep export tool for Austrian/EU service companies (MVP focused on BOTFORCE GmbH). It prepares monthly handoff packages for an external accountant and is explicitly not a full accounting system.

### Scope (In)

- Authentication & role-based access (superadmin, employee, accountant) with strict Row-Level Security (RLS)
- Customers, projects with assignments, time tracking workflow, expense workflow with receipts
- Invoices and credit notes with Austrian/EU compliance (sequential numbering, immutable after issue)
- PDF generation, email delivery, payment reminders
- Monthly accounting export (CSV + PDFs + receipts in ZIP)
- Audit logging

### Out of Scope (Explicit)

- Full double-entry accounting (general ledger, chart of accounts, automated VAT returns)
- Bank synchronization/integration (Revolut or other banks) for MVP
- Automated recurring invoice scheduler (can be added later)

## 2. Comparison & Reconciliation (Source A vs Source B)

### Alignment

- Same product scope: invoicing + time + expenses + accounting export packages (not full accounting).
- Same core tech: Next.js 14 App Router + TypeScript + Tailwind + Supabase + Vercel.
- Same roles and workflows; same Austrian/EU compliance concepts.

### Differences

- Source A is a detailed requirements blueprint (to-be).
- Source B is an implementation/runbook style document (as-built and operational).
- Source B adds repo structure, setup instructions, testing tools, and known limitations.

### Conflicts to Resolve (Canonical Decisions)

- Project assignments: DB support may be ready; admin UI is pending — treat as Partial until UI shipped.
- Expense category enum: finalize canonical list and migrate/mapping accordingly.
- Signed URL TTL: enforce standard TTL (target 1 hour) and document it.
- Time recording mode start/end: confirm & implement, or remove from MVP.

## 3. Master Specification v1.0 (Canonical Single Source of Truth)

### 3.1 Tech Stack (Canonical)

- Frontend: Next.js 14+ (App Router), TypeScript (strict), Tailwind CSS
- Backend: Supabase (Postgres, Auth, Storage, RLS)
- PDF: jsPDF + jspdf-autotable

- Email: Resend
- ZIP: JSZip; Dates: date-fns; Icons: Lucide React
- Deploy: Vercel (frontend) + Supabase Cloud (backend); VCS: GitHub
- Testing: Vitest + Playwright (E2E)

### 3.2 Security & Compliance (Canonical)

- RLS on all tables; multi-tenant company_id isolation
- Immutable records after issue/invoiced; document snapshots at issue time
- VAT: 20% standard, 10% reduced, 0% zero-rated; Reverse charge for EU B2B
- Sequential numbering per company/type/year with thread-safe generation
- Receipts stored securely; signed URLs with expiring tokens (target TTL 1 hour)
- Audit log for critical actions and state transitions

### 3.3 Roles & Permissions (Canonical)

- superadmin: full access, approvals, team/company management, exports
- employee: own time & expenses; only assigned projects
- accountant: read finance + create exports; no deletions/modifications

### 3.4 Modules (Canonical)

- Auth & Authorization
- Company, Team, Customers, Projects + Assignments
- Time tracking workflow; Expense workflow with receipts
- Documents (invoice/credit note): drafts, numbering, issue, PDF, email
- Accounting export: preview, generate ZIP, lock export
- Finance dashboard (AR/AP + cash forecast)
- Recurring invoice templates (manual generate; scheduler later)
- Audit logging

### 3.5 Canonical Data Model (Summary)

- companies, profiles, company_members, customers, projects, project_assignments, time_entries, expenses, files, documents, document_lines, document_number_series, accounting_exports, accounting_export_lines, recurring_invoice_templates, recurring_invoice_lines, audit_log.

### 3.6 Canonical Page Structure

- /login, /reset-password, /(authenticated)/dashboard, customers, projects, timesheets, documents (+ recurring), expenses, finance, team, settings

## 4. Sprint Plan & User Story Backlog (Implementation via Claude Code)

### 4.1 Working Agreements (for Claude Code)

- Claude Code produces PR-style diffs (file paths + patches).
- All DB changes via Supabase migrations (ordered SQL).
- Never expose SUPABASE_SERVICE_ROLE_KEY to the browser.
- All state transitions must be enforced server-side (DB triggers/functions and server actions).

- All monetary amounts DECIMAL(15,2); all IDs UUID; locale de-AT formatting.
- Definition of Done: implemented + typecheck + lint + basic tests + RLS verified.

## Sprint 0: Foundation & Tooling

Goal: Create a reproducible repo, Supabase baseline, and UI design system.

### Deliverables

- Repo structure (apps/web + supabase) with pnpm workspaces
- Tailwind dark theme tokens + base UI components
- Supabase baseline migrations: profiles, companies, company_members; RLS enabled; seed data

### User Stories

| ID | Role | User Story | Acceptance Criteria (summary) |
|---|---|---|---|
| US-000 | Developer | Repo scaffolding + scripts | pnpm dev works; lint/typecheck scripts; README updated |
| US-001 | All users | Dark theme tokens + base UI | Cards/buttons/inputs follow tokens; responsive layout |
| US-002 | Developer | Supabase baseline + seed users | Migrations apply; 3 test users; RLS enabled |

### Claude Code Prompt Packet (paste into Claude Code)

```
You are Claude Code working in a Next.js 14 + Supabase monorepo.
Implement Sprint 0 (US-000..US-002).
Rules: output a PR-style diff; add SQL migrations under supabase/migrations; no service role in client; create Tailwind tokens; add README setup steps; ensure TypeScript strict passes.
```

## Sprint 1: Auth, Permissions & Navigation

Goal: Ship authentication flow, protected routes, role checks, and navigation shell.

### Deliverables

- Login/logout/reset password flow
- Protected routes and role-based route protection
- Authenticated layout with role-aware navigation

### User Stories

| ID | Role | User Story | Acceptance Criteria (summary) |
|---|---|---|---|
| US-010 | All roles | Login/logout/reset password | Auth works; protected routes redirect; reset flow works |
| US-011 | All roles | Role-aware navigation | Menu matches role; 403 for unauthorized |

| | | | |
|---|---|---|---|
| US-012 | Developer | Permission map + helpers | Single source for role permissions; reusable guards |

### Claude Code Prompt Packet (paste into Claude Code)

```
Implement Sprint 1 (US-010..US-012). Use Supabase Auth correctly with App Router. Add middleware/guards for protected routes and role-based access. Build authenticated layout with nav items filtered by role.
```

## Sprint 2: Company & Team Management

Goal: Enable company profile setup and manage team members with roles, rates, activation.

### Deliverables

- Company profile & settings page
- Team page: list members, invite, change role, set rate, deactivate/reactivate
- Audit entries for team and company changes

### User Stories

| ID | Role | User Story | Acceptance Criteria (summary) |
|---|---|---|---|
| US-020 | Superadmin | Company profile CRUD | Edit legal data + logo; validation; audit logged |
| US-021 | Superadmin | Invite member by email | Invite flow; role assigned; audit logged |
| US-022 | Superadmin | Change role + hourly rate | Role/rate changes persist; safety (min 1 admin) |
| US-023 | Superadmin | Deactivate/reactivate member | is_active enforced; history preserved |

### Claude Code Prompt Packet (paste into Claude Code)

```
Implement Sprint 2 (US-020..US-023): companies + company_members policies and UI. Add invite flow (server action) and team management UI. Enforce RLS and safety: cannot remove last superadmin; deactivated users cannot create new records.
```

## Sprint 3: Customers

Goal: Create customer base with reverse-charge and compliance-ready fields.

### Deliverables

- Customer CRUD + archive + filtering
- Reverse charge auto-detection rule
- Audit logging for customer changes

### User Stories

| ID | Role | User Story | Acceptance Criteria (summary) |
|---|---|---|---|
| US-030 | Superadmin | Customer CRUD | Create/edit; list filter/search; archive supported |
| US-031 | System | Reverse charge auto-detection | EU non-AT + VAT → reverse_charge=true; used in invoicing |
| US-032 | Superadmin | Delete rules | Hard delete only if no documents exist; otherwise archive |

### Claude Code Prompt Packet (paste into Claude Code)

```
Implement Sprint 3 (US-030..US-032): customers table, RLS, server actions, UI pages.
Implement reverse charge auto-detection and store flags. Add audit logging.
```

## Sprint 4: Projects & Assignments

Goal: Enable project CRUD, budgets, and assignment UI (critical for employee access control).

### Deliverables

- Project CRUD (hourly/fixed, budgets, active/billable)
- Project assignment UI with rate overrides
- Employee access limited to assigned projects

### User Stories

| ID | Role | User Story | Acceptance Criteria (summary) |
|---|---|---|---|
| US-040 | Superadmin | Project CRUD | Create/edit/archive; filter by customer/status |
| US-041 | Superadmin | Assignments UI | Assign/unassign employees; optional rate override |
| US-042 | Employee | Assigned projects visibility | Employees only see assigned projects everywhere |

### Claude Code Prompt Packet (paste into Claude Code)

```
Implement Sprint 4 (US-040..US-042): projects + project_assignments schema, RLS, UI.
Ensure employee access is constrained to assigned projects, including time entry creation.
```

## Sprint 5: Time Tracking Workflow

Goal: Implement time entry creation, submission, approvals, and invoicing linkage.

### Deliverables

- Time entry CRUD for employees (draft/rejected only)
- Submission and approval queue (bulk approve/reject)
- Server-enforced status transitions; rate snapshot
- Optional: start/end mode (decide and implement)

### User Stories

| ID | Role | User Story | Acceptance Criteria (summary) |
|---|---|---|---|
| US-050 | Employee | Create/edit time entries | Draft editable; only assigned projects; filters/totals |
| US-051 | Employee | Submit entries | draft→submitted; timestamps; audit logged |
| US-052 | Superadmin | Approve/reject (bulk) with reasons | Queue; approve/reject sets by/at; bulk actions |
| US-053 | System | Rate snapshot + immutability | Capture rate on approval; invoiced entries immutable |
| US-054 | Decision | Start/end mode support | Either implemented end-to-end or removed from MVP |

### Claude Code Prompt Packet (paste into Claude Code)

```
Implement Sprint 5 (US-050..US-054): time_entries schema, RLS, status machine (DB
trigger), UI list/editor, approval queue with bulk actions, and rate snapshot resolution
(project rate + assignment override + member default).
```

## Sprint 6: Expenses & Receipts

Goal: Implement expenses with receipts, approval workflow, and invoice/export readiness.

### Deliverables

- Expense CRUD + receipt upload to Supabase Storage
- Expense workflow with approvals (bulk)
- Mileage (0.42 EUR/km default) and travel-time helpers
- Signed URL access (TTL target 1h)

### User Stories

| ID | Role | User Story | Acceptance Criteria (summary) |
|---|---|---|---|
| US-060 | Employee | Create/edit expenses + upload receipt | Draft editable; receipt stored; secure access |
| US-061 | Employee | Mileage & travel time calculations | Calculated fields; configurable default km rate |

| | | | |
|---|---|---|---|
| US-062 | Superadmin | Expense approvals (bulk) | Queue; approve/reject; audit; exported status later |
| US-063 | Decision | Finalize expense category enum | Canonical enum decided and applied consistently |

### Claude Code Prompt Packet (paste into Claude Code)

```
Implement Sprint 6 (US-060..US-063): expenses + files schema, secure storage upload,
signed URL retrieval, expense workflow and approval queue with bulk actions. Finalize and
enforce a canonical category enum.
```

## Sprint 7: Invoicing, Numbering, PDFs

Goal: Create and issue invoices/credit notes with compliant numbering, snapshots, locking and PDFs.

### Deliverables

- Document drafts with line items (manual + from time/expenses)
- Thread-safe numbering series per company/type/year
- Issue: snapshots + locking + update linked time/expenses
- PDF generation (reverse charge notice DE+EN)

### User Stories

| ID | Role | User Story | Acceptance Criteria (summary) |
|---|---|---|---|
| US-070 | Superadmin | Create document drafts + lines | No number in draft; edit; add lines/manual |
| US-071 | Superadmin | Add approved time/expenses to invoices | Select items; create lines; store linkage IDs |
| US-072 | System | Issue with numbering + snapshots + lock | Sequential number; JSON snapshots; immutable lines |
| US-073 | System | Generate PDF | PDF layout correct; reverse charge notice when needed |
| US-074 | Superadmin | Mark paid / cancel | Paid date set; cancel rules defined; audit logged |

### Claude Code Prompt Packet (paste into Claude Code)

```
Implement Sprint 7 (US-070..US-074): documents + document_lines + series, issue
transaction, locking, PDF generation with jsPDF-autotable, and status operations
(paid/cancel) with audit.
```

## Sprint 8: Email + Accounting Export

Goal: Enable sending invoices/reminders and generating monthly export ZIP packages.

### Deliverables

- Resend integration: send invoice + payment reminders
- Accounting export preview (period selection)
- Generate ZIP package: CSV + PDFs + receipts; lock export; download

### User Stories

| ID | Role | User Story | Acceptance Criteria (summary) |
|---|---|---|---|
| US-080 | Superadmin | Send invoice email | Resend sends; PDF attached or secure link; audit logged |
| US-081 | Superadmin | Send payment reminder | Overdue reminders; audit logged |
| US-082 | Accountant | Export preview | Shows included docs/expenses; totals and counts |
| US-083 | Accountant | Generate export ZIP + lock | ZIP contains csv/pdfs/receipts; export locks records |
| US-084 | System | Signed URL standardization | TTL=1h default; secure access enforced |

### Claude Code Prompt Packet (paste into Claude Code)

```
Implement Sprint 8 (US-080..US-084): Resend email actions and templates; accounting export tables and UI; ZIP generation with JSZip; CSV output; include PDFs and receipts; lock export; enforce signed URL TTL and permissions.
```

## Sprint 9: Finance Dashboard + Recurring + Hardening

Goal: Deliver AR/AP widgets, cash forecast MVP, recurring templates MVP, and production hardening.

### Deliverables

- Finance dashboard: AR aging + AP pending reimbursements + monthly summary
- Cash forecast MVP (starting balance + recurring costs)
- Recurring templates CRUD + manual generate
- Hardening: audit coverage, indexing/pagination, tests + CI

### User Stories

| ID | Role | User Story | Acceptance Criteria (summary) |
|---|---|---|---|
| US-090 | Accountant | AR aging buckets | Total AR, overdue AR, aging buckets; list overdue invoices |
| US-091 | Superadmin | AP pending | List approved not |

| | | | |
|---|---|---|---|
| | | reimbursements + mark reimbursed | reimbursed; action sets reimbursed_at |
| US-092 | Superadmin | Cash forecast (12-week) | Inflow from due invoices; outflow from recurring costs + reimbursements |
| US-093 | Superadmin | Recurring templates + manual generate | CRUD templates; generate draft invoice |
| US-094 | Developer | Audit coverage + triggers | All critical actions logged; tests for audit completeness |
| US-095 | Developer | Indexes/pagination | Fast list pages; key indexes; pagination implemented |
| US-096 | Developer | Tests + CI | Vitest + Playwright; CI runs lint/typecheck/tests |

#### Claude Code Prompt Packet (paste into Claude Code)

```
Implement Sprint 9 (US-090..US-096): finance dashboard queries and UI, cash forecast MVP, recurring templates UI, and hardening tasks: audit trigger coverage, indexes + pagination, test suite + CI workflow.
```

## 5. Claude Code Playbook

### 5.1 Recommended Prompt Template

```
You are Claude Code working inside an existing Next.js 14 + Supabase monorepo.
Implement Sprint <N> stories: <IDs>.
Deliver PR-style diffs only.
Add DB changes as SQL migrations (supabase/migrations).
Enforce RLS (company_id isolation) and role permissions.
Never expose SUPABASE_SERVICE_ROLE_KEY to client code.
Add/Update tests where meaningful.
Ensure pnpm lint + pnpm typecheck pass.
```

### 5.2 Quality Checklist

- All mutations happen via Server Actions or secure API endpoints
- Issue actions are transactional and idempotent where needed
- Audit logs capture old/new payloads for critical entities
- PDFs are deterministic (same inputs produce same output)
- Exports are immutable once locked

## Appendix A — Source A (Verbatim)

Complete Requirements Specification (as provided):

```
# BOTFORCE Unity - Complete Requirements Specification

## Project Overview

Build a comprehensive business management platform for Austrian/EU service companies. The
application handles time tracking, expense management, invoicing, and accounting exports
with full Austrian tax compliance.

**Tech Stack:**
- Frontend: Next.js 14+ (App Router), TypeScript, Tailwind CSS
- Backend: Supabase (PostgreSQL, Auth, Storage, Row-Level Security)
- Deployment: Vercel
- Version Control: GitHub
- Design: Dark theme matching www.botforce-discovery.com

---

## NON-FUNCTIONAL REQUIREMENTS

### 1. Technology Stack

```
Framework:        Next.js 14+ with App Router
Language:         TypeScript (strict mode)
Styling:          Tailwind CSS with custom dark theme
Database:         Supabase PostgreSQL
Authentication:   Supabase Auth (email/password)
File Storage:     Supabase Storage
State:            React Server Components + Server Actions
PDF Generation:   jsPDF
ZIP Creation:     JSZip
Date Handling:    date-fns
Icons:            Lucide React
Deployment:       Vercel
```
### 2. Design System

```
Background:       Dark navy (#0a0e1a to #141928)
Cards:            rgba(255, 255, 255, 0.04) with rgba(255,255,255,0.08) borders
Primary Button:   #1f5bff (bright blue)
Success:          #22c55e (green)
Warning:          #f59e0b (amber)
Danger:           #ef4444 (red)
Text Primary:     #ffffff
Text Secondary:   rgba(232, 236, 255, 0.68)
Text Muted:       rgba(232, 236, 255, 0.5)
Border Radius:    18px (cards), 12px (buttons), 10px (inputs)
Font Sizes:       11px (labels), 13px (body), 14-15px (headings), 24px (page titles)
```
### 3. Security Requirements

- Row-Level Security (RLS) on all database tables
- Company-level data isolation (multi-tenant)
```

- Secure file URLs with expiring signed tokens (1 hour)
- Audit logging for all data modifications
- Document locking after issuance (immutable records)
- Service role key for admin operations only

### 4. Performance Requirements

- Server-side rendering for initial page loads
- Optimistic UI updates where appropriate
- Indexed database queries for common operations
- Lazy loading for large data sets
- Revalidation on data mutations

### 5. Compliance Requirements (Austrian/EU)

- VAT rates: 20% standard, 10% reduced, 0% exempt
- Sequential invoice numbering per company/year
- Immutable invoice snapshots (customer + company data frozen at issue)
- Reverse charge handling for EU B2B transactions
- Kilometergeld support (0.42 EUR/km default)
- Receipt attachment for expense documentation

---

## FUNCTIONAL REQUIREMENTS

### Module 1: Authentication & Authorization

#### 1.1 User Authentication
- Email/password login via Supabase Auth
- Password reset functionality
- Session management with automatic refresh
- Redirect to login for unauthenticated requests

#### 1.2 User Roles
```
superadmin:  Full access - manage company, team, all features
employee:    Limited - own time entries, own expenses, assigned projects
accountant:  Read-only finance - view all, create exports, no modifications
```

#### 1.3 Authorization Rules
- Superadmin: CRUD all entities, manage team, approve workflows
- Employee: Create/edit own drafts, submit for approval, view assigned projects
- Accountant: Read all financial data, create exports, no write operations

---

### Module 2: Company Management

#### 2.1 Company Profile
```
Fields:
- name (required)
- legal_name (required)
- vat_number (ATU format for Austria)
- registration_number (FN format)
```

- address_line1, address_line2
- postal_code, city, country
- email, phone, website
- logo_url
- settings (JSON for future extensibility)
```

#### 2.2 Company Settings
- Default payment terms
- Invoice prefix configuration
- Tax settings

---

### Module 3: Team Management

#### 3.1 Team Members
```
Fields:
- user_id (link to auth.users via profiles)
- company_id
- role (superadmin, employee, accountant)
- hourly_rate (default rate for time entries)
- is_active (soft delete support)
- invited_at, joined_at
```

#### 3.2 Team Operations
- Invite new members by email
- Assign/change roles
- Set hourly rates per member
- Deactivate members (preserve history)
- Reactivate previously deactivated members
- Send invitation emails with role-specific content

---

### Module 4: Customer Management

#### 4.1 Customer Profile
```
Fields:
- company_id (tenant isolation)
- name, legal_name
- vat_number
- tax_exempt (boolean)
- reverse_charge (auto-set for EU non-AT countries)
- email, phone
- address_line1, address_line2, postal_code, city, country
- payment_terms_days (default: 14)
- default_tax_rate
- currency (default: EUR)
- notes
- is_active
```

#### 4.2 Customer Operations
```

- Create/edit customers
- Archive customers (soft delete)
- Delete customers (only if no documents exist)
- Auto-detect reverse charge based on country
- List customers with filtering

---

### Module 5: Project Management

#### 5.1 Project Configuration
```
Fields:
- company_id, customer_id
- name, code (unique identifier like ACME-WEB)
- description
- billing_type: 'hourly' | 'fixed'
- hourly_rate (for hourly billing)
- fixed_price (for fixed billing)
- budget_hours, budget_amount
- start_date, end_date
- time_recording_mode: 'hours' | 'start_end'
- is_active, is_billable
```

#### 5.2 Project Assignments
```
Fields:
- project_id, user_id
- hourly_rate_override (optional, overrides project rate)
- assigned_at, unassigned_at
- is_active
```

#### 5.3 Project Operations
- Create/edit projects
- Assign/unassign team members
- Set per-member rate overrides
- Archive projects
- View project hours and budget consumption
- Filter by customer, status

---

### Module 6: Time Tracking

#### 6.1 Time Entry
```
Fields:
- company_id, project_id, user_id
- date
- hours (direct entry mode)
- start_time, end_time, break_minutes (start/end mode)
- description
- is_billable
- hourly_rate (captured on approval)
- status: 'draft' | 'submitted' | 'approved' | 'rejected' | 'invoiced'
```

```
- submitted_at, approved_at, approved_by
- rejected_at, rejected_by, rejection_reason
- document_id (when invoiced)
- invoiced_at
```

#### 6.2 Time Entry Workflow
```
Draft → Submit → Approved → Invoiced
              ↓
           Rejected → (edit) → Resubmit
```

#### 6.3 Time Entry Operations
- Create entries (employees: own only)
- Edit draft/rejected entries
- Submit for approval
- Approve/reject with reason (superadmin only)
- Bulk approve/reject
- Filter by date range, project, user, status
- Group by day with collapsible view
- Calculate totals by period

#### 6.4 Time Recording Modes
- **Hours mode:** Enter decimal hours directly (e.g., 7.5)
- **Start/End mode:** Enter start time, end time, break minutes; auto-calculate hours

---

### Module 7: Expense Management

#### 7.1 Expense Entry
```
Fields:
- company_id, user_id, project_id (optional)
- date
- amount (gross amount including tax)
- currency (EUR)
- tax_rate: 'standard_20' | 'reduced_10' | 'zero'
- tax_amount (auto-calculated)
- category: 'mileage' | 'travel_time' | 'materials' | 'accommodation' | 'meals' |
'transport' | 'communication' | 'software' | 'other'
- description
- merchant
- receipt_file_id (link to files table)
- is_reimbursable
- reimbursed_at
- status: 'draft' | 'submitted' | 'approved' | 'rejected' | 'exported'
- submitted_at, approved_at, approved_by
- rejected_at, rejected_by, rejection_reason
- exported_at, export_id
```

#### 7.2 Expense Categories
```
mileage:        Kilometergeld (0.42 EUR/km default)
travel_time:    Travel time compensation (hourly)
```

```
materials:      Project materials
accommodation:  Hotels, lodging
meals:          Food and beverages
transport:      Public transport, taxi, flights
communication:  Phone, internet
software:       Software licenses
other:          Miscellaneous expenses
```

#### 7.3 Expense Workflow
```
Draft → Submit → Approved → Exported
              ↓
           Rejected → (edit) → Resubmit
```

#### 7.4 Expense Operations
- Create expenses with receipt upload
- Edit draft/rejected expenses
- Submit for approval
- Approve/reject with reason (superadmin only)
- Bulk approve/reject
- Add approved expenses to invoices as line items
- Filter by date range, category, status, user
- View receipt files

---

### Module 8: Document Management (Invoicing)

#### 8.1 Document Types
```
invoice:      Standard invoice (INV prefix)
credit_note:  Credit memo (CN prefix)
```

#### 8.2 Document Structure
```
Header:
- company_id, customer_id
- document_type, document_number (auto-generated)
- status: 'draft' | 'issued' | 'paid' | 'cancelled'
- issue_date, due_date, paid_date
- customer_snapshot (JSON - frozen at issue)
- company_snapshot (JSON - frozen at issue)
- payment_terms_days
- payment_reference (bank transfer reference)
- notes (visible on invoice)
- internal_notes (internal only)
- is_locked, locked_at

Totals:
- subtotal (sum of line subtotals)
- tax_amount (sum of line taxes)
- total (subtotal + tax_amount)
- tax_breakdown (JSON - amounts per tax rate)
- currency
```

```
Lines:
- line_number (order)
- description
- quantity, unit (hours, pcs, km, etc.)
- unit_price
- tax_rate: 'standard_20' | 'reduced_10' | 'zero' | 'reverse_charge'
- subtotal (qty * unit_price)
- tax_amount
- total
- time_entry_ids (array - linked time entries)
- project_id (optional)
```

#### 8.3 Document Numbering
```
Format: {PREFIX}-{YEAR}-{PADDED_NUMBER}
Example: INV-2026-00001, CN-2026-00001

Rules:
- Sequential per company, document type, year
- Thread-safe generation (database lock)
- Auto-increment on issue
- Configurable prefix and padding
```

#### 8.4 Document Workflow
```
Draft → Issue → Paid
            ↓
         Cancelled
```

#### 8.5 Document Operations
- Create draft invoice/credit note
- Add line items (manual or from time entries/expenses)
- Edit draft documents
- Issue document (locks, assigns number, creates snapshots)
- Mark as paid
- Cancel issued document
- Generate PDF
- Send via email
- Send payment reminder
- Delete draft documents only
- Filter by type, status, customer, date range

#### 8.6 PDF Generation
```
Invoice PDF includes:
- Company header (logo, name, address, VAT, registration)
- Customer details (name, address, VAT)
- Invoice details (number, date, due date)
- Line items table (description, qty, unit, price, tax, total)
- Tax breakdown summary
- Grand total
- Payment terms and reference
- Notes
```

```

---

### Module 9: Recurring Invoices

#### 9.1 Recurring Template
```
Fields:
- company_id, customer_id
- name, description
- frequency: 'weekly' | 'biweekly' | 'monthly' | 'quarterly' | 'yearly'
- day_of_month (for monthly/quarterly/yearly)
- day_of_week (for weekly/biweekly)
- payment_terms_days
- notes
- is_active
- next_issue_date
- last_issued_at
- subtotal, tax_amount, total

Template Lines:
- description, quantity, unit, unit_price, tax_rate
- project_id (optional)
```

#### 9.2 Recurring Operations
- Create/edit templates
- Activate/deactivate templates
- Generate invoice from template (manual)
- Auto-generate on schedule (future: cron job)
- Delete templates

---

### Module 10: Financial Dashboard

#### 10.1 Accounts Receivable
```
Metrics:
- Total AR (sum of unpaid invoices)
- Overdue AR (past due date)
- Aging buckets: Current, 1-30 days, 31-60 days, 61+ days
- List of unpaid invoices with customer, amount, days overdue
```

#### 10.2 Accounts Payable
```
Metrics:
- Pending reimbursements (approved expenses not yet reimbursed)
- List of pending expenses with employee, amount, date
```

#### 10.3 Cash Flow Forecast
```
12-week projection:
- Week start date

```
- Expected inflows (invoices due in that week)
- Expected outflows (recurring costs + pending reimbursements)
- Weekly net
- Cumulative balance

Inputs:
- Starting cash balance (user input)
- Recurring costs (list of weekly/monthly/quarterly/yearly costs)
```

#### 10.4 Monthly Summary
```
- Revenue this month (paid invoices)
- Expenses this month (approved expenses)
- Recent invoices list
- Recent expenses list
```

---

### Module 11: Accounting Export

#### 11.1 Export Configuration
```
Fields:
- company_id
- name, description
- period_start, period_end (month boundaries)
- status: 'pending' | 'processing' | 'completed' | 'failed'
- created_by
- processed_at, completed_at, failed_at
- error_message
- csv_file_id, zip_file_id
- invoice_count, credit_note_count, expense_count
- total_revenue, total_expenses
- is_locked, locked_at
```

#### 11.2 Export Contents
```
CSV columns:
- Type (Invoice, Credit Note, Expense)
- Document Number
- Date
- Customer/Vendor
- Category (for expenses)
- Amount (EUR)
- Tax Rate
- Tax Amount
- Status

ZIP package:
/accounting_export_January_2026.zip
├── summary.csv
├── invoices/
│   ├── INV-2026-00001.pdf
│   └── INV-2026-00002.pdf
```

```
└── receipts/
    ├── 2026-01-15_mileage_42.00EUR.jpg
    └── 2026-01-20_materials_150.00EUR.pdf
```

#### 11.3 Export Operations
- Preview export (show what will be included)
- Create export (generates CSV, marks expenses as exported)
- Download ZIP package (CSV + PDFs + receipts)
- Lock export (prevents modifications to included records)

---

### Module 12: Audit Logging

#### 12.1 Audit Entry
```
Fields:
- company_id, user_id, user_email
- action: 'create' | 'update' | 'delete' | 'status_change' | 'issue' | 'approve' |
'reject' | 'lock'
- table_name
- record_id
- old_data (JSON)
- new_data (JSON)
- metadata (JSON - additional context)
- ip_address, user_agent
- created_at
```

#### 12.2 Logged Events
- Document created, updated, issued, paid, cancelled
- Time entry created, submitted, approved, rejected, invoiced
- Expense created, submitted, approved, rejected, exported
- Customer created, updated, archived
- Project created, updated, archived
- Team member invited, role changed, deactivated
- Export created, locked

---

### Module 13: Dashboard

#### 13.1 Dashboard Widgets
```
For Superadmin:
- Total hours this week/month
- Pending approvals (time entries + expenses)
- Outstanding invoices count and amount
- Revenue this month
- Recent activity feed

For Employee:
- My hours this week
- My pending submissions
- My assigned projects
- Recent time entries
```

For Accountant:
- Outstanding AR
- Pending expenses
- Recent exports
- Monthly revenue/expenses
```

---

### Module 14: Settings

#### 14.1 Company Settings
- Edit company profile
- Configure invoice numbering
- Set default payment terms
- Upload company logo

#### 14.2 User Profile
- Edit name, phone
- Change password
- Upload avatar

---

## DATA MODEL SUMMARY

```
companies (1) ──────┬──── (*) company_members ──────── (1) profiles
                    │
                    ├──── (*) customers
                    │           │
                    │           └──── (*) documents ──── (*) document_lines
                    │                                          │
                    ├──── (*) projects ──────┤
                    │           │            │
                    │           └──── (*) project_assignments
                    │                                          │
                    ├──── (*) time_entries ──┘
                    │
                    ├──── (*) expenses ──── (0..1) files
                    │
                    ├──── (*) accounting_exports ──── (*) accounting_export_lines
                    │
                    ├──── (*) recurring_invoice_templates ──── (*) recurring_invoice_lines
                    │
                    ├──── (*) document_number_series
                    │
                    ├──── (*) files
                    │
                    └──── (*) audit_log
```

---

## API ENDPOINTS (Server Actions)
```

```
### Auth Actions
- `signIn(email, password)`
- `signOut()`
- `resetPassword(email)`

### Company Actions
- `updateCompany(data)`
- `getCompany()`

### Team Actions
- `inviteTeamMember(email, role)`
- `updateMemberRole(userId, role)`
- `updateMemberRate(userId, rate)`
- `removeMember(userId)`
- `getTeamMembers()`

### Customer Actions
- `createCustomer(data)`
- `updateCustomer(id, data)`
- `deleteCustomer(id)`
- `getCustomers()`
- `getCustomer(id)`

### Project Actions
- `createProject(data)`
- `updateProject(id, data)`
- `deleteProject(id)`
- `assignMemberToProject(projectId, userId, rateOverride?)`
- `removeMemberFromProject(projectId, userId)`
- `getProjects()`
- `getProject(id)`
- `getAvailableTeamMembers(projectId)`

### Time Entry Actions
- `createTimeEntry(data)`
- `updateTimeEntry(id, data)`
- `deleteTimeEntry(id)`
- `submitTimeEntry(id)`
- `approveTimeEntry(id)`
- `rejectTimeEntry(id, reason)`
- `bulkApproveTimeEntries(ids)`
- `bulkRejectTimeEntries(ids, reason)`
- `getTimeEntries(filters)`
- `getApprovalQueue()`

### Expense Actions
- `createExpense(data)`
- `updateExpense(id, data)`
- `deleteExpense(id)`
- `submitExpense(id)`
- `approveExpense(id)`
- `rejectExpense(id, reason)`
- `bulkApproveExpenses(ids)`
- `bulkRejectExpenses(ids, reason)`
- `uploadReceipt(expenseId, file)`
- `getExpenses(filters)`
- `getApprovalQueue()`
```

### Document Actions
- `createDocument(type, customerId)`
- `updateDocument(id, data)`
- `deleteDocument(id)`
- `addDocumentLine(documentId, data)`
- `updateDocumentLine(lineId, data)`
- `deleteDocumentLine(lineId)`
- `issueDocument(id)`
- `markDocumentPaid(id, paidDate)`
- `cancelDocument(id)`
- `generateDocumentPDF(id)`
- `sendDocumentEmail(id, recipientEmail?)`
- `sendPaymentReminder(id)`
- `addTimeEntriesToDocument(documentId, timeEntryIds)`
- `addExpensesToDocument(documentId, expenseIds)`
- `getDocuments(filters)`
- `getDocument(id)`

### Recurring Invoice Actions
- `createRecurringTemplate(data)`
- `updateRecurringTemplate(id, data)`
- `deleteRecurringTemplate(id)`
- `toggleTemplateActive(id)`
- `generateInvoiceFromTemplate(id)`
- `getRecurringTemplates()`

### Finance Actions
- `getFinancialSummary(startingBalance, recurringCosts)`
- `getExportPreview(year, month)`
- `createAccountingExport(year, month, name, description?)`
- `downloadExportPackage(exportId)`
- `getAccountingExports()`

### File Actions
- `uploadFile(file, category)`
- `getSignedUrl(fileId)`
- `deleteFile(fileId)`

---

## PAGE STRUCTURE

```
/                          → Redirect to /dashboard
/login                     → Login page
/reset-password           → Password reset

/(authenticated)/
├── dashboard/             → Main dashboard
├── timesheets/
│   ├── page              → Time entry list with filters
│   ├── new/              → Create time entry
│   └── [id]/edit/        → Edit time entry
├── projects/
│   ├── page              → Project list
│   ├── new/              → Create project
```

```
│   └── [id]/
│       ├── page          → Project detail
│       └── edit/         → Edit project
├── customers/
│   ├── page              → Customer list
│   ├── new/              → Create customer
│   └── [id]/
│       ├── page          → Customer detail
│       └── edit/         → Edit customer
├── documents/
│   ├── page              → Invoice/credit note list
│   ├── new/              → Create document
│   ├── [id]/
│   │   ├── page          → Document detail
│   │   └── edit/         → Edit document
│   └── recurring/
│       ├── page          → Recurring template list
│       ├── new/          → Create template
│       └── [id]/edit/    → Edit template
├── expenses/
│   ├── page              → Expense list
│   ├── new/              → Create expense
│   └── [id]/edit/        → Edit expense
├── finance/
│   └── page              → Financial dashboard + exports
├── team/
│   └── page              → Team management
└── settings/
    └── page              → Company and user settings
```

---

## IMPLEMENTATION PRIORITIES

### Phase 1: Foundation
1. Supabase setup (database, auth, storage, RLS policies)
2. Next.js project with auth flow
3. Company and profile management
4. Basic layout and navigation

### Phase 2: Core Operations
5. Customer management
6. Project management with assignments
7. Time entry tracking and workflow
8. Expense management with receipts

### Phase 3: Invoicing
9. Document creation and line items
10. Document numbering and issuance
11. PDF generation
12. Email delivery

### Phase 4: Finance
13. Financial dashboard
14. Accounting exports
15. Recurring invoices

### Phase 5: Polish
16. Audit logging
17. Bulk operations
18. Performance optimization
19. Error handling and validation

---

## TESTING CHECKLIST

### Functional Tests
- [ ] User can login/logout
- [ ] User can create/edit/delete customers
- [ ] User can create/edit/delete projects
- [ ] User can assign team members to projects
- [ ] User can create/submit time entries
- [ ] Admin can approve/reject time entries
- [ ] User can create/submit expenses with receipts
- [ ] Admin can approve/reject expenses
- [ ] Admin can create/issue invoices
- [ ] Invoice numbers are sequential
- [ ] PDF generation works correctly
- [ ] Email delivery works
- [ ] Accounting export includes all data
- [ ] RLS prevents cross-company data access

### Compliance Tests
- [ ] Tax calculations are correct (20%, 10%, 0%)
- [ ] Reverse charge applies for EU non-AT customers
- [ ] Invoice snapshots are immutable after issue
- [ ] Document cannot be deleted after issue
- [ ] Audit log captures all modifications
- [ ] Kilometergeld calculates correctly

---

## NOTES

- All monetary amounts stored as DECIMAL(15,2)
- All dates stored as DATE or TIMESTAMPTZ
- All UUIDs use uuid_generate_v4()
- All tables have created_at and updated_at timestamps
- Soft delete preferred (is_active flag) to preserve referential integrity
- Use Austrian locale for date/currency formatting (de-AT)
- Default currency is EUR throughout.

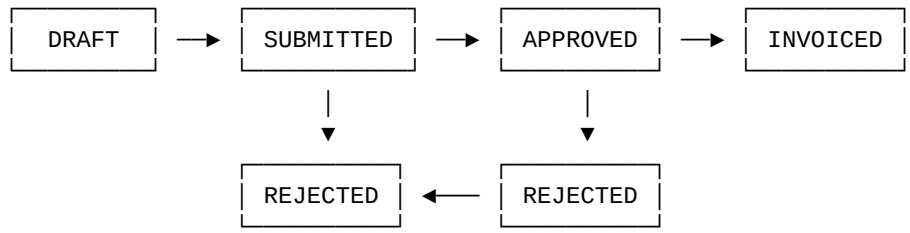
## Appendix B — Source B (Verbatim)

Product / Implementation Documentation (as provided):

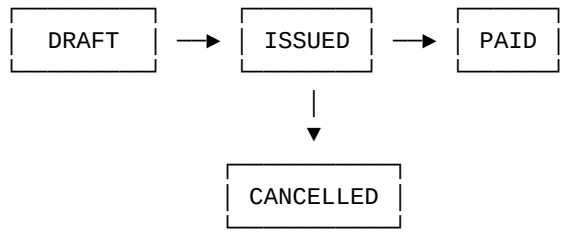
```
BOTFORCE Unity
A production-ready invoicing, time tracking, expense management, and accounting-prep tool
for BOTFORCE GmbH (Austria, Vienna).

Note: This is NOT a full accounting system. It's an invoicing + time tracking + expenses +
accounting-prep export tool designed to prepare monthly packages for handoff to an
external accountant.

Features
Core Functionality
Feature      Status Description
Multi-Role Access   ✅ Complete  Superadmin, Employee, Accountant with strict RLS
Project Management  ✅ Complete  Projects under customers with employee assignments
Time Tracking ✅ Complete  Full workflow: draft → submitted → approved → invoiced
Austrian Invoicing  ✅ Complete  Sequential numbers, immutable after issue, PDF generation
Expense Management  ✅ Complete  Receipt upload, approval workflow, mileage/travel time
Accounting Export   ✅ Complete  CSV + PDF ZIP packages for accountant handoff
Reverse Charge (EU B2B)    ✅ Complete  Auto-detect for EU customers, PDF notice
Email Integration   ✅ Complete  Invoice sending, payment reminders
User Roles
Role   Capabilities
SUPERADMIN    Full access: all projects, time entries, expenses, documents, team management
EMPLOYEE      View assigned projects only, manage own time entries (draft/submitted), own
expenses
ACCOUNTANT    Read-only on documents/expenses, create accounting exports, no deletions
Time Entry Workflow
```




```
Employees: Can move draft → submitted
Superadmins: Can approve/reject (with reason), bulk operations supported
Invoiced: Immutable, locked with rate snapshot
Document Workflow
```




```
Draft: Editable, no document number
Issued: Locked, sequential number assigned, customer/company snapshots stored
Paid: Payment date recorded
Cancelled: Voided (typically with credit note)
Tech Stack
Frontend: Next.js 14 (App Router) + TypeScript + Tailwind CSS
Backend: Supabase (Postgres, Auth, Storage, RLS, Edge Functions)
PDF Generation: jsPDF with jspdf-autotable
```

Email: Resend
Deployment: Vercel (frontend) + Supabase Cloud (backend)
Testing: Vitest + Playwright
Project Structure

```
botforce-unity/
├── apps/
│   └── web/                      # Next.js application
│       ├── src/
│       │   ├── app/              # App Router pages
│       │   │   ├── (authenticated)/  # Protected routes
│       │   │   │   ├── dashboard/
│       │   │   │   ├── customers/
│       │   │   │   ├── projects/
│       │   │   │   ├── timesheets/
│       │   │   │   ├── documents/
│       │   │   │   ├── expenses/
│       │   │   │   ├── finance/
│       │   │   │   ├── accounting-export/
│       │   │   │   ├── team/
│       │   │   │   └── settings/
│       │   │   ├── actions/      # Server actions
│       │   │   └── login/
│       │   ├── components/       # React components
│       │   │   ├── ui/           # Base UI components
│       │   │   ├── team/         # Team management
│       │   │   └── expenses/     # Expense components
│       │   ├── lib/              # Utilities
│       │   │   ├── supabase/     # Supabase clients
│       │   │   ├── pdf/          # PDF generation
│       │   │   └── email/        # Email templates
│       │   └── types/            # TypeScript types
│       └── e2e/                  # Playwright tests
├── supabase/
│   ├── migrations/               # SQL migrations (ordered)
│   └── seed/                     # Development seed data
└── .github/
    └── workflows/                # CI/CD pipelines
```

Getting Started
Prerequisites
Node.js 18+
pnpm 8+
Supabase CLI
Docker (for local Supabase)
Local Development
Clone the repository

```
git clone https://github.com/botforce-team/botforce-unity.git
cd botforce-unity
```

Install dependencies

```
pnpm install
```

Start Supabase locally

```
supabase start
```

Run migrations and seed data

```
supabase db reset
```

Set up environment variables

Create apps/web/.env.local:

```
NEXT_PUBLIC_SUPABASE_URL=http://localhost:54321
NEXT_PUBLIC_SUPABASE_ANON_KEY=<your-anon-key>
SUPABASE_SERVICE_ROLE_KEY=<your-service-role-key>

# Email (optional for local dev)
RESEND_API_KEY=<your-resend-key>
```

Start the development server

```
pnpm dev
```

Open the app Navigate to http://localhost:3000

Default Test Users (Local Development)

After running supabase db reset, these users are available:

| Email | Password | Role |
|---|---|---|
| admin@botforce.at | password123 | Superadmin |
| employee@botforce.at | password123 | Employee |
| accountant@botforce.at | password123 | Accountant |

Database Schema

Core Tables

| Table | Description |
|---|---|
| companies | Tenant companies (single-tenant MVP, multi-tenant ready) |
| profiles | User profiles linked to auth.users |
| company_members | User-company-role assignments |
| customers | Client companies with address, VAT, reverse charge settings |
| projects | Projects under customers with billing type, rates, budgets |
| project_assignments | Employee-project access (many-to-many) |
| time_entries | Time logs with workflow status and rate snapshots |
| documents | Invoices and credit notes with immutability |
| document_lines | Invoice line items with per-line tax rates |
| document_number_series | Sequential numbering per company/type/year |
| expenses | Expense records with approval workflow |
| files | File metadata for receipts and attachments |
| accounting_exports | Monthly export packages with statistics |
| accounting_export_lines | Export line items tracking |
| audit_log | Immutable action audit trail |

Row Level Security (RLS)

All tables have RLS enabled with company_id isolation:

Employees: Can only access assigned projects and own time entries

Superadmins: Full access to all company data

Accountants: Read-only on documents/expenses, can create exports

Key Database Features

Sequential Document Numbers: Thread-safe function with row-level locking

Customer/Company Snapshots: JSONB storage at issue time for immutability

Time Entry Workflow: Trigger-enforced status transitions

Automatic Calculations: Tax amounts, totals, hours from start/end times

Audit Trail: All critical actions logged with user snapshots

Austrian Invoicing Compliance

Document Numbering

Format: PREFIX-YEAR-NUMBER (e.g., INV-2026-00001)

Sequential per company, document type, and year

Thread-safe generation with row-level locking

Tax Rates
Rate Value Description
standard_20 20% Standard Austrian VAT
reduced_10 10% Reduced rate (food, etc.)
zero 0% Zero-rated (exports, B2B services)
Reverse Charge (EU B2B)
Automatic detection based on customer country and VAT number
PDF includes required notice in German and English:
"VAT reverse charge: The recipient of the service is liable for VAT. Steuerschuldnerschaft des Leistungsempfängers gem. Art. 196 Richtlinie 2006/112/EG."

Immutability
Documents locked after issue (only status/payment fields editable)
Customer and company details snapshotted at issue time
Line items cannot be modified after issue
Expense Categories
Category Description Calculation
mileage Kilometergeld Distance × €0.42/km (Austrian standard)
travel_time Reisezeit Hours × Hourly Rate
reimbursement Auslagenersatz Direct amount entry
Accounting Export
The accounting export feature generates monthly packages for accountant handoff:

Contents
CSV Summary: All invoices, credit notes, and expenses with:

Document numbers and dates
Customer/vendor names
Net amounts, tax rates, tax amounts
Status information
PDF Invoices: All issued invoices for the period

Receipt Attachments: Expense receipts (when attached)

Export Workflow
Navigate to Finance → Accounting Export
Select date range
Preview included items
Generate and download ZIP package
Testing
Run Tests
# Unit tests
pnpm test

# E2E tests
pnpm test:e2e

# Type checking
pnpm typecheck
RLS Policy Tests
# Reset database with seed data
supabase db reset

# Run RLS tests
psql $DATABASE_URL -f tests/rls_tests.sql
Deployment
Vercel (Frontend)

Import repository in Vercel
Set environment variables:
NEXT_PUBLIC_SUPABASE_URL
NEXT_PUBLIC_SUPABASE_ANON_KEY
SUPABASE_SERVICE_ROLE_KEY
RESEND_API_KEY
Deploy
Supabase (Backend)
Create project at supabase.com
Link project:
supabase link --project-ref your-project-ref
Push migrations:
supabase db push
Configure Storage buckets for receipts
API Reference
Server Actions

| Action | File | Description |
|---|---|---|
| createTimeEntry | time-entries.ts | Create new time entry |
| submitTimeEntry | time-entries.ts | Submit for approval |
| approveTimeEntry | time-entries.ts | Approve entry (admin) |
| bulkApproveTimeEntries | time-entries.ts | Bulk approval |
| createDocument | documents.ts | Create invoice/credit note |
| issueDocument | documents.ts | Issue and lock document |
| generateDocumentPDF | documents.ts | Generate PDF |
| sendDocumentByEmail | documents.ts | Email invoice |
| createExpense | expenses.ts | Create expense |
| approveExpense | expenses.ts | Approve expense (admin) |
| uploadReceipt | expenses.ts | Upload receipt file |
| createAccountingExport | finance.ts | Generate export package |
| getFinancialSummary | finance.ts | Get financial dashboard data |

Known Limitations
Single Company: MVP designed for single company (BOTFORCE GmbH), but schema supports multi-tenant
Bank Integration: Not implemented (out of scope for MVP)
Recurring Invoices: Database schema ready, frontend pending
Project Assignment UI: Database supports it, admin UI pending
Contributing
Create feature branch from main
Make changes with tests
Run pnpm lint and pnpm typecheck
Submit PR for review
License
Proprietary - BOTFORCE GmbH