Thought Leadership Engine - Sprint 1

Authentication System & Brand Voice Foundations

Sprint Duration: 10 days (2 weeks - Mon-Fri)
Sprint Goal: Users can register, log in securely, and create their first brand profile with voice calibration
Team Capacity: 5 developers × 6 hours/day × 10 days = 300 hours
Story Points: 55 SP (targeting 50-60% of historical velocity for first sprint)

🎯 Sprint Objectives

Primary Goals

1. ✅ Complete authentication system (JWT + 2FA)
2. ✅ Brand profile CRUD operations
3. ✅ Voice profile configuration UI
4. ✅ Basic dashboard shell

Success Metrics

- [ ] Users can register and log in
- [ ] 2FA can be enabled
- [ ] Brand profiles can be created
- [ ] Voice tone can be configured
- [ ] Test coverage >80%
- [ ] All APIs documented in Postman

📋 Sprint Backlog

Epic 1: Authentication System (25 SP)

Story 1.1: User Registration (8 SP)

As a new user
I want to create an account
So that I can access the platform

Acceptance Criteria:

- [ ] Registration form with email, password, first name, last name
- [ ] Password must be 8+ characters with special char
- [ ] Email validation (format + uniqueness)
- [ ] Passwords hashed with bcrypt (12 rounds)
- [ ] Confirmation email sent
- [ ] User record created in MongoDB

Technical Tasks:

- [ ] Create User schema with validation
- [ ] POST /api/v1/auth/register endpoint
- [ ] Password hashing middleware
- [ ] Email uniqueness check
- [ ] Unit tests for registration logic
- [ ] Integration test for full flow

Owner: Renee
Reviewer: Sentinel
Estimated: 2 days

Story 1.2: User Login (5 SP)

As a registered user
I want to log in with email/password
So that I can access my account

Acceptance Criteria:

- [ ] Login form with email and password
- [ ] Returns JWT access token (1h expiry)
- [ ] Returns refresh token (7d expiry)
- [ ] Invalid credentials returns 401
- [ ] Rate limited to 5 attempts per 15 min
- [ ] Last login timestamp updated

Technical Tasks:

- [ ] POST /api/v1/auth/login endpoint

- [ ] JWT token generation
- [ ] Password comparison logic
- [ ] Refresh token storage (Redis)
- [ ] Rate limiting middleware
- [ ] Login form component

Owner: Renee
Reviewer: Sentinel
Estimated: 1.5 days

Story 1.3: Token Refresh (3 SP)

As a logged-in user
I want my session to extend automatically
So that I don't get logged out while working

Acceptance Criteria:

- [ ] POST /api/v1/auth/refresh endpoint
- [ ] Accepts refresh token
- [ ] Returns new access token
- [ ] Validates refresh token not expired/revoked
- [ ] Frontend auto-refreshes before expiry

Technical Tasks:

- [ ] Refresh endpoint implementation
- [ ] Refresh token validation
- [ ] Frontend axios interceptor for auto-refresh
- [ ] Redis check for revoked tokens
- [ ] Unit tests

Owner: Sammi
Reviewer: Renee
Estimated: 1 day

Story 1.4: Two-Factor Authentication (9 SP)

As a security-conscious user
I want to enable 2FA
So that my account is more secure

Acceptance Criteria:

- [ ] Enable 2FA button in settings
- [ ] QR code generated with TOTP secret
- [ ] Verification code input (6 digits)
- [ ] 2FA required on login if enabled
- [ ] Backup codes generated (10)
- [ ] Can disable 2FA with verification

Technical Tasks:

- [ ] Install speakeasy library
- [ ] POST /api/v1/auth/2fa/enable endpoint
- [ ] QR code generation
- [ ] TOTP verification logic
- [ ] Backup codes generation + storage
- [ ] 2FA settings UI component
- [ ] Login flow modification for 2FA
- [ ] POST /api/v1/auth/2fa/verify endpoint

Owner: Sentinel
Reviewer: Renee
Estimated: 2.5 days

Epic 2: Brand Profile System (20 SP)

Story 2.1: Create Brand Profile (8 SP)

As a user
I want to create a brand profile
So that I can start generating content

Acceptance Criteria:

- [ ] Brand creation form with name, archetype
- [ ] 12 archetype options with descriptions
- [ ] POST /api/v1/brand endpoint
- [ ] User can have multiple brands
- [ ] Brand associated with user account
- [ ] Validation prevents duplicate names per user

Technical Tasks:

- [ ] BrandProfile schema implementation

- [ ] POST /api/v1/brand endpoint
- [ ] Brand creation form UI
- [ ] Archetype selector component with tooltips
- [ ] Form validation (client + server)
- [ ] Unit tests
- [ ] E2E test: Create first brand

Owner: Lexa
Reviewer: Renee
Estimated: 2 days

Story 2.2: Voice Profile Configuration (12 SP)

As a brand owner
I want to configure my brand voice
So that content matches my tone

Acceptance Criteria:

- [ ] Voice configuration wizard (3 steps)
- [ ] Step 1: Select tone tags (multi-select)
- [ ] Step 2: Choose vocabulary preferences
- [ ] Step 3: Set emotional range (slider 1-10)
- [ ] Preview voice description generated
- [ ] Save voice profile to brand
- [ ] Can edit voice profile later

Technical Tasks:

- [ ] Voice profile sub-schema
- [ ] PUT /api/v1/brand/:id/voice endpoint
- [ ] Tone tag selector (checkboxes)
- [ ] Emotional range slider component
- [ ] Vocabulary input (tag input)
- [ ] Voice preview generator
- [ ] BrandVoiceEngine.getVoiceContext() implementation
- [ ] Validation logic

Owner: Lexa + Alex Z
Reviewer: Renee
Estimated: 3 days

Epic 3: Dashboard Shell (10 SP)

Story 3.1: Dashboard Layout (5 SP)

As a user
I want to see a dashboard after login
So that I have a home base

Acceptance Criteria:

- [ ] Sidebar navigation (brands, content, personas, campaigns)
- [ ] Header with user menu
- [ ] Brand selector dropdown
- [ ] Responsive layout (mobile, tablet, desktop)
- [ ] Dark theme applied
- [ ] Loading states for data fetching

Technical Tasks:

- [ ] Dashboard layout component
- [ ] Sidebar component with navigation
- [ ] Header component with user dropdown
- [ ] Brand context provider (React Context)
- [ ] Route protection (require auth)
- [ ] Responsive CSS with Tailwind
- [ ] Loading skeleton components

Owner: Alex Z
Reviewer: Renee
Estimated: 1.5 days

Story 3.2: Basic Metrics Cards (5 SP)

As a user
I want to see key metrics on my dashboard
So that I understand my progress

Acceptance Criteria:

- [ ] 4 metric cards: Content, Views, Conversions, Revenue
- [ ] Mock data for Sprint 1 (real data in Sprint 4)
- [ ] Bento grid layout
- [ ] Animated number counting
- [ ] Trend indicators (+/-)

- [ ] Responsive card layout

Technical Tasks:

- [ ] MetricCard component
- [ ] Bento grid CSS layout
- [ ] Number animation hook
- [ ] Trend calculation util
- [ ] Mock data service
- [ ] Responsive breakpoints

Owner: Alex Z + Finlay
Reviewer: Renee
Estimated: 1.5 days

📅 Sprint 1 Schedule

Week 1

Day	Focus	Ceremonies
Mon	Sprint Planning	Planning (2h), Setup stories
Tue	Auth - Registration	Daily Standup (15min)
Wed	Auth - Login + Refresh	Daily Standup, Mid-sprint check-in
Thu	Auth - 2FA Start	Daily Standup
Fri	Auth - 2FA Complete	Daily Standup, Demo (internal)

Week 2

Day	Focus	Ceremonies
Mon	Brand Profile Creation	Daily Standup
Tue	Voice Configuration	Daily Standup
Wed	Dashboard Shell	Daily Standup, Mid-sprint check-in
Thu	Integration & Testing	Daily Standup
Fri	Sprint Review & Retro	Standup, Review (1h), Retro (1h)

🏗️ Technical Architecture for Sprint 1

Backend Structure

```
src/
├── models/
│   ├── User.ts (NEW)
│   └── BrandProfile.ts (NEW)
├── routes/
│   ├── auth.routes.ts (NEW)
│   └── brand.routes.ts (NEW)
├── controllers/
│   ├── auth.controller.ts (NEW)
│   └── brand.controller.ts (NEW)
├── services/
│   ├── auth.service.ts (NEW)
│   ├── token.service.ts (NEW)
│   └── brandVoice.service.ts (NEW)
├── middleware/
│   ├── authenticate.ts (NEW)
│   ├── validate.ts (NEW)
│   └── rateLimiter.ts (NEW)
├── utils/
│   ├── jwt.ts (NEW)
│   └── password.ts (NEW)
└── tests/
    ├── auth.test.ts (NEW)
    └── brand.test.ts (NEW)
```

Frontend Structure

```
src/
├── pages/
│   ├── Auth/
│   │   ├── Login.tsx (NEW)
│   │   ├── Register.tsx (NEW)
│   │   └── TwoFactor.tsx (NEW)
│   ├── Dashboard/
│   │   └── Dashboard.tsx (NEW)
│   └── Brand/
│       ├── CreateBrand.tsx (NEW)
│       └── VoiceConfiguration.tsx (NEW)
```

```
├── components/
│   ├── Layout/
│   │   ├── DashboardLayout.tsx (NEW)
│   │   ├── Sidebar.tsx (NEW)
│   │   └── Header.tsx (NEW)
│   ├── Brand/
│   │   ├── ArchetypeSelector.tsx (NEW)
│   │   └── VoiceWizard.tsx (NEW)
│   └── Common/
│       ├── MetricCard.tsx (NEW)
│       └── LoadingSkeleton.tsx (NEW)
├── services/
│   ├── api.ts (NEW)
│   └── auth.ts (NEW)
├── hooks/
│   ├── useAuth.ts (NEW)
│   └── useBrand.ts (NEW)
├── context/
│   ├── AuthContext.tsx (NEW)
│   └── BrandContext.tsx (NEW)
└── utils/
    └── validation.ts (NEW)
```

🔄 Daily Workflow

Daily Standup Format (15 min max)

Each person answers:
1. What I completed yesterday
2. What I'm working on today
3. Any blockers

Renee facilitates, takes notes, assigns unblocking tasks

Pull Request Process

1. Create feature branch: feature/sprint-1-story-X
2. Implement story with tests
3. Self-review checklist:
   - [ ] Tests pass locally
   - [ ] Lint passes
   - [ ] No console.logs

  - [ ] Code formatted
  - [ ] Types correct (TS)
4. Push and create PR
5. Request review from designated reviewer
6. Address feedback
7. Merge when approved + CI passes

Code Review SLA

- **Initial review:** Within 4 hours
- **Follow-up review:** Within 2 hours
- **Emergency:** <1 hour (ask in Slack)

🧪 Testing Requirements

Unit Test Coverage Targets

- **Controllers:** 90%
- **Services:** 95%
- **Utils:** 100%
- **Components:** 80%

Test Examples

Backend Unit Test:

```
// tests/services/auth.service.test.ts
describe('AuthService', () => {
  describe('register', () => {
    it('should create user with hashed password', async () => {
      const input = {
        email: 'test@example.com',
        password: 'SecurePass123!',
        firstName: 'Test',
        lastName: 'User'
      };

      const user = await authService.register(input);

      expect(user.email).toBe(input.email);
      expect(user.passwordHash).not.toBe(input.password);
```

```
    expect(await bcrypt.compare(input.password, user.passwordHash)).toBe(true);
  });

  it('should reject duplicate email', async () => {
    await authService.register(existingUser);

    await expect(
      authService.register(existingUser)
    ).rejects.toThrow('Email already exists');
  });
 });
});
```

Frontend Component Test:

```
// tests/components/Login.test.tsx
describe('Login Component', () => {
 it('should submit login form with valid credentials', async () => {
  const mockLogin = jest.fn();
  render(<Login onLogin={mockLogin} />);

  await userEvent.type(screen.getByLabelText('Email'), 'test@example.com');
  await userEvent.type(screen.getByLabelText('Password'), 'password123');
  await userEvent.click(screen.getByRole('button', { name: 'Log In' }));

  expect(mockLogin).toHaveBeenCalledWith({
   email: 'test@example.com',
   password: 'password123'
  });
 });

 it('should show validation errors for empty fields', async () => {
  render(<Login />);

  await userEvent.click(screen.getByRole('button', { name: 'Log In' }));

  expect(screen.getByText('Email is required')).toBeInTheDocument();
  expect(screen.getByText('Password is required')).toBeInTheDocument();
 });
});
```

Integration Test:

```
// tests/integration/auth.test.ts
describe('Auth Flow Integration', () => {
  it('should complete full registration and login flow', async () => {
    // Register
    const registerRes = await request(app)
      .post('/api/v1/auth/register')
      .send({
        email: 'newuser@test.com',
        password: 'SecurePass123!',
        firstName: 'New',
        lastName: 'User'
      });

    expect(registerRes.status).toBe(201);

    // Login
    const loginRes = await request(app)
      .post('/api/v1/auth/login')
      .send({
        email: 'newuser@test.com',
        password: 'SecurePass123!'
      });

    expect(loginRes.status).toBe(200);
    expect(loginRes.body).toHaveProperty('accessToken');
    expect(loginRes.body).toHaveProperty('refreshToken');

    // Access protected route
    const profileRes = await request(app)
      .get('/api/v1/user/profile')
      .set('Authorization', `Bearer ${loginRes.body.accessToken}`);

    expect(profileRes.status).toBe(200);
    expect(profileRes.body.email).toBe('newuser@test.com');
  });
});
```

📊 Definition of Done

Story-Level DoD

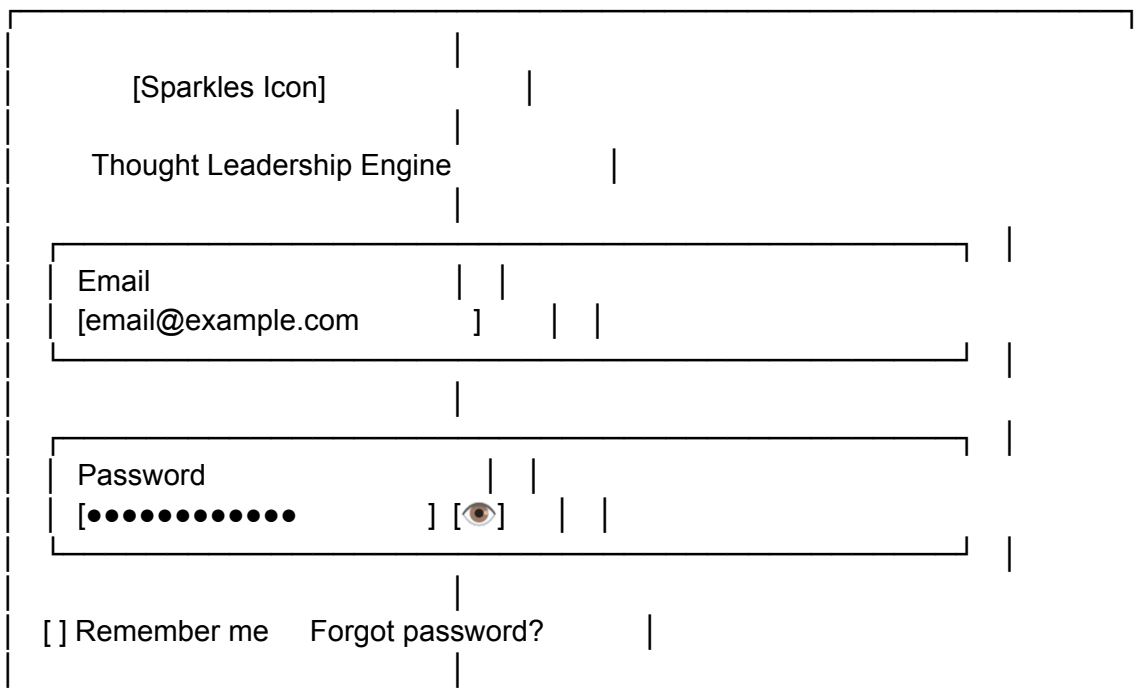
- [ ] Acceptance criteria met
- [ ] Code reviewed and approved
- [ ] Unit tests written (>80% coverage)
- [ ] Integration tests added (if API change)
- [ ] Manual testing completed
- [ ] Documentation updated
- [ ] No new linting errors
- [ ] Deployed to staging
- [ ] Product owner accepts

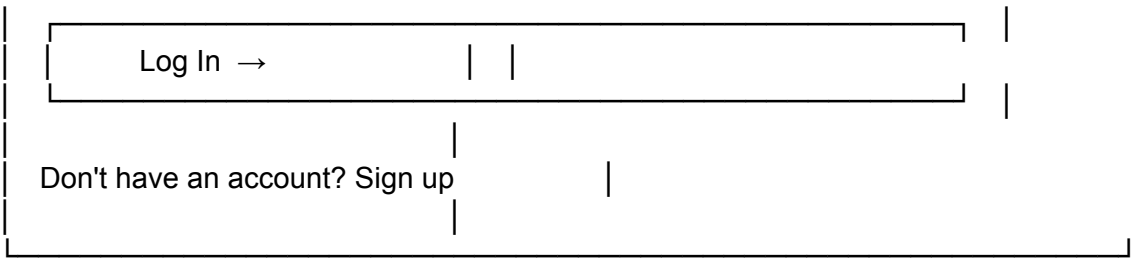
Sprint-Level DoD

- [ ] All stories meet story DoD
- [ ] Sprint goal achieved
- [ ] No P0/P1 bugs open
- [ ] All code merged to develop
- [ ] Staging environment stable
- [ ] Sprint demo completed
- [ ] Retrospective action items documented

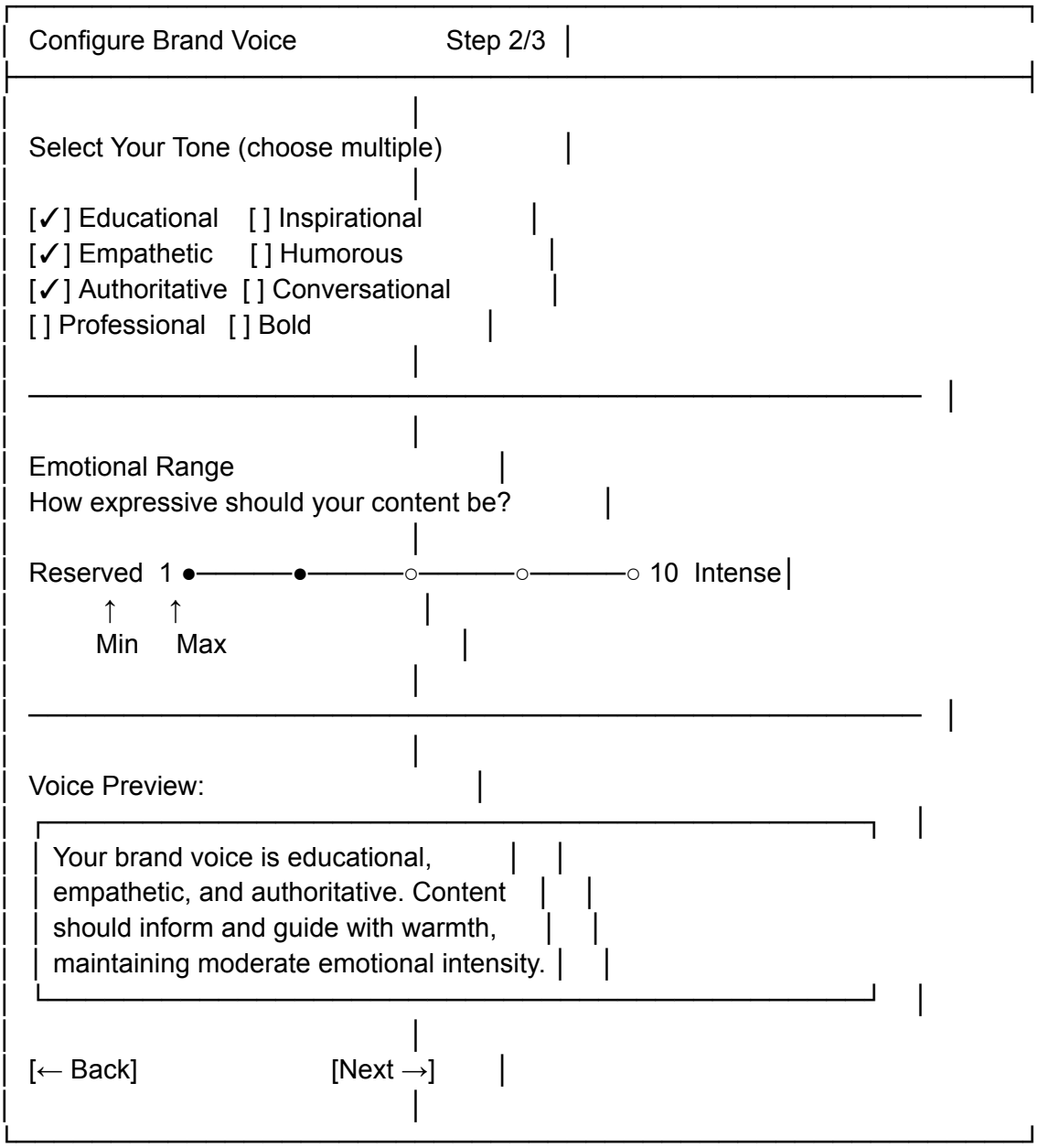
🎨 UI/UX Design Specifications

Login Page Mockup

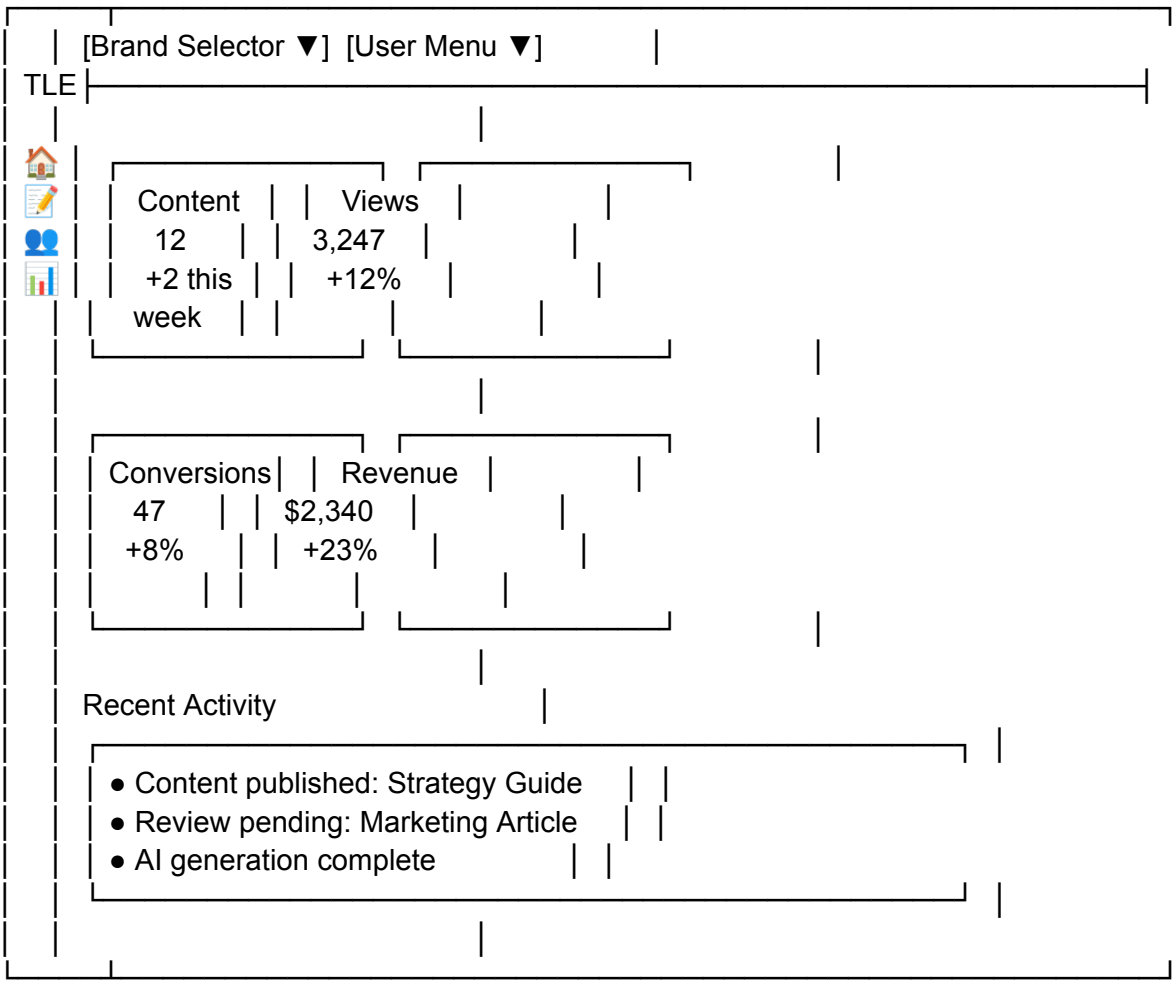




Voice Configuration Wizard



Dashboard Shell



🚀 Deployment Strategy

Continuous Deployment Flow

Developer Push
  ↓
GitHub PR Created
  ↓
CI Pipeline Runs
├─ Lint Check
├─ Unit Tests
├─ Build Check
└─ Security Scan
  ↓

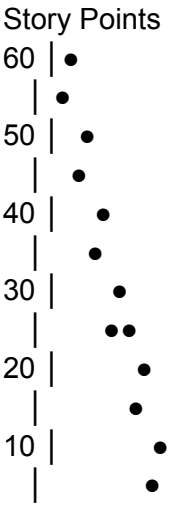
PR Approved
  ↓
Merge to Develop
  ↓
Auto-deploy to Staging
├─ Railway (Backend)
└─ Vercel (Frontend)
  ↓
Manual QA on Staging
  ↓
Merge Develop → Main
  ↓
Auto-deploy to Production
  ↓
Smoke Tests Run
  ↓
Monitoring Alerts Active

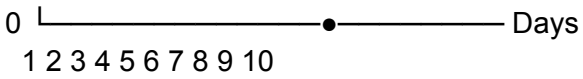
Staging URLs

- Frontend: https://staging-tle.vercel.app
- Backend API: https://tle-api-staging.up.railway.app
- Docs: https://docs-staging.tle.dev

📈 Sprint Metrics & Tracking

Burndown Chart Target





Velocity Tracking

Sprint 1 Target: 55 SP
- Completed: TBD
- Carry-over: TBD
- Velocity: TBD

(Baseline for Sprint 2 planning)

Risk Register

Risk	Probability	Impact	Mitigation
2FA complexity underestimated	Medium	High	Pair programming, early spike
Team member sick	Low	Medium	Cross-training, documentation
Third-party API issues	Low	High	Fallback to email-based 2FA
Scope creep	Medium	Medium	Strict story acceptance, PO approval

---

🎯 Sprint Review Agenda (Friday Week 2)

Attendees

- Development Team
- Product Owner (if applicable)
- Stakeholders

Agenda (1 hour)

1. **Sprint Goal Review** (5 min)
    - Did we achieve the sprint goal?
2. **Demo** (30 min)
    - Story 1.1: User Registration

    - Story 1.2: Login Flow
    - Story 1.3: Token Refresh (show dev tools)
    - Story 1.4: 2FA Enable/Verify
    - Story 2.1: Create Brand Profile
    - Story 2.2: Voice Configuration Wizard
    - Story 3.1: Dashboard Layout
    - Story 3.2: Metric Cards
3. **Metrics Review** (10 min)
    - Velocity achieved
    - Test coverage
    - Code quality
4. **Feedback** (10 min)
    - What worked well?
    - What could be improved?
    - Questions from stakeholders
5. **Sprint 2 Preview** (5 min)
    - Next sprint goals
    - Dependencies

🔄 Sprint Retrospective Agenda

Format: Sailboat Retrospective

☀️ Wind (What moved us forward?)
⚓ Anchor (What held us back?)
🪨 Rocks (What risks ahead?)
🏝️ Island (What's our goal?)

Questions

1. **What gave us wind?** (Keep doing)

2. **What was our anchor?** (Stop doing)
3. **What rocks ahead?** (Risks for Sprint 2)
4. **Are we headed to the island?** (On track?)

Action Items Template

Action Items from Sprint 1 Retro

1. [Action]: Improve PR review time
   - Owner: Renee
   - Due: Before Sprint 2
   - Status: 🟢 In Progress

2. [Action]: Add more UI component examples
   - Owner: Alex Z
   - Due: Day 2 Sprint 2
   - Status: 🔴 Not Started

✅ Sprint 1 Checklist

Pre-Sprint

- [ ] Sprint planning complete
- [ ] Stories estimated
- [ ] Stories assigned
- [ ] Acceptance criteria clear
- [ ] Design mockups ready
- [ ] Dependencies identified

During Sprint

- [ ] Daily standups happening
- [ ] Burndown chart updated
- [ ] Blockers resolved quickly
- [ ] Code reviews timely
- [ ] Tests passing
- [ ] Staging deployments working

End of Sprint

- [ ] All stories demo-ready
- [ ] Sprint review completed

- [ ] Retrospective completed
- [ ] Sprint 2 backlog refined
- [ ] Velocity recorded
- [ ] Lessons learned documented

📞 Support & Communication

Daily Standup: 9:30 AM

Mid-Sprint Check-in: Wed 2:00 PM

Sprint Review: Fri Week 2, 2:00 PM

Retrospective: Fri Week 2, 3:30 PM

Slack Channels

- #sprint-1 - Sprint-specific discussion
- #development - Technical questions
- #design - UI/UX feedback
- #blockers - Urgent issues

Pair Programming

- Available daily 10-12 PM, 2-4 PM
- Book via #development
- Recommended for Story 1.4 (2FA) and Story 2.2 (Voice Config)

🎉 Sprint 1 Success Criteria

Sprint 1 is successful when:

✅ **A new user can:**

1. Register an account
2. Log in and see dashboard
3. Enable 2FA
4. Create a brand profile
5. Configure brand voice

6. See placeholder metrics

✅ **The system provides:**

- Secure authentication
- Token-based sessions
- Protected API routes
- Responsive UI
- 80%+ test coverage

✅ **The team has:**

- Established velocity baseline
- Smooth collaboration
- Clear next steps

Sprint 1 Kickoff: Monday Week 1, 9:00 AM
Sprint 1 Complete: Friday Week 2, 5:00 PM
Sprint 2 Planning: Monday Week 3, 9:00 AM

Let's ship it! 🚀