# Nexia（ネクシア）ウェブアプリケーション 要件定義書

## 1. プロジェクト概要

### 1.1 プロジェクト名

Nexia（ネクシア）- デジタル追悼・家系つながりサービス

### 1.2 プロジェクトの目的

高齢者の死後の悩み解決を主目的とし、故人の生きた証をデジタルで未来へつなぎ、心の安らぎと新たなつながりを生み出すデジタル追悼プラットフォームを開発する。

### 1.3 対象ユーザー

- 主要ターゲット：高齢者（60歳以上）とその家族
- 副次ターゲット：お墓の管理に悩む方、遠方に住む親族、自分の生きた証を残したい方

### 1.4 4つの主要目的

1. 自分が亡くなるときの悩みを解決すること
2. 遠い先祖とのつながりを作ること
3. 生きているとき・若いときから後世に残すものを考えること（各自にとっての鏡になること）
4. 親戚やその他の属性（同じ考え方、同じ地域、学校など）による横のつながりの拡大

## 2. 機能要件

### 2.1 ユーザー管理機能

#### 2.1.1 ユーザー登録・認証

- メールアドレスとパスワードによる新規ユーザー登録
- ログイン・ログアウト機能
- パスワードリセット機能
- プロフィール情報の管理（氏名、生年月日、住所、電話番号等）
- アカウント削除機能

### 2.1.2 権限管理

- 一般ユーザー、プレミアムユーザーの区別
- 管理者権限の設定
- 家族・親戚間での権限共有機能

## 2.2 デジタル墓碑・追悼ページ機能

### 2.2.1 故人ページの作成・編集

- 故人の基本情報入力（氏名、生年月日、没年月日、出身地等）
- プロフィール写真のアップロード
- 生前の写真・動画・音声ファイルのアップロード
- 手記・エピソード・思い出の文章入力
- 趣味・特技・職歴等の詳細情報入力

### 2.2.2 公開範囲設定

- 本人限定（作成者のみ閲覧可能）
- 家族限定（指定した家族メンバーのみ閲覧可能）
- 友人限定（指定した友人のみ閲覧可能）
- 一般公開（誰でも閲覧可能）
- コンテンツごとの個別公開設定

### 2.2.3 メモリアル機能

- 故人への追悼メッセージ投稿
- バーチャル献花機能
- 線香・ろうそくのバーチャル供養
- 命日・誕生日等の記念日通知機能

## 2.3 バーチャルお参り機能

### 2.3.1 オンライン参拝

- 故人ページでのバーチャル参拝
- 参拝履歴の記録
- 参拝時のメッセージ投稿

- 参拝者リストの表示

### 2.3.2 オンライン法要

- 提携寺院との連携機能
- オンライン読経サービスの予約・参加
- 法要スケジュールの管理
- 法要参加者の管理

## 2.4 デジタル家系図機能

### 2.4.1 家系図作成・編集

- 家族関係の登録（親子、夫婦、兄弟姉妹等）
- 家系図の視覚的表示
- 複数世代にわたる家系図の管理
- 家系図の共同編集機能

### 2.4.2 家族情報管理

- 各家族メンバーの詳細情報
- 家族写真・エピソードの共有
- 家族イベント・記念日の管理
- 家族間のメッセージ機能

## 2.5 ライフログ・未来への手紙機能

### 2.5.1 ライフログ記録

- 日記・思考の記録機能
- 写真・動画付きライフログ
- 人生の重要な出来事の記録
- タイムライン表示機能

### 2.5.2 未来への手紙

- 将来の家族・子孫へのメッセージ作成
- 指定日時での自動公開設定

- 条件付き公開機能（例：自分の死後に公開）
- 教訓・価値観の記録

### 2.6 コミュニティ・つながり機能

#### 2.6.1 親戚コミュニティ

- 共通の先祖を持つ親戚グループの作成
- グループ内での情報共有
- グループイベントの企画・管理
- グループチャット機能

#### 2.6.2 属性別コミュニティ

- 同じ地域・学校・趣味等のコミュニティ
- 公開・非公開コミュニティの選択
- コミュニティ掲示板機能
- イベント告知・参加機能

### 2.7 物理的連携機能

#### 2.7.1 QRコード連携

- 故人ページ用QRコードの生成
- QRコードのカスタマイズ機能
- QRコードアクセス履歴の管理
- 印刷用QRコードの出力

#### 2.7.2 DNA情報連携（オプション）

- 提携サービスとのAPI連携
- DNA情報の安全な保管
- 血縁関係の可視化
- 遺伝的特徴の記録

## 3. 非機能要件

### 3.1 セキュリティ要件

### 3.1.1 データ保護

- 個人情報の暗号化保存
- HTTPS通信の必須化
- パスワードのハッシュ化
- 定期的なセキュリティ監査

### 3.1.2 アクセス制御

- 多要素認証の実装
- セッション管理の強化
- 不正アクセス検知機能
- ログイン試行回数制限

### 3.1.3 プライバシー保護

- GDPR準拠のデータ処理
- ユーザーによるデータ削除権
- データ利用目的の明確化
- 第三者提供の制限

## 3.2 パフォーマンス要件

### 3.2.1 応答時間

- ページ読み込み時間：3秒以内
- API応答時間：1秒以内
- 画像・動画アップロード：進捗表示付き
- 検索機能：2秒以内

### 3.2.2 スケーラビリティ

- 同時接続ユーザー数：1,000人以上
- データベース容量：拡張可能な設計
- CDN活用による高速配信
- 負荷分散の実装

## 3.3 可用性要件

### 3.3.1 稼働率

- システム稼働率：99.9%以上
- 定期メンテナンス時間の最小化
- 障害時の自動復旧機能
- バックアップ・復旧体制

### 3.3.2 災害対策

- データの地理的分散保存
- 災害時の事業継続計画
- 定期的なバックアップ実行
- 復旧手順の文書化

## 3.4 ユーザビリティ要件

### 3.4.1 操作性

- 直感的なユーザーインターフェース
- 高齢者にも使いやすい設計
- 多言語対応（日本語、英語）
- アクセシビリティ対応

### 3.4.2 デバイス対応

- レスポンシブデザイン
- スマートフォン・タブレット対応
- 主要ブラウザ対応
- PWA（Progressive Web App）対応

## 3.5 運用・保守要件

### 3.5.1 監視・ログ

- システム監視の自動化
- エラーログの収集・分析

- ユーザー行動の分析
- パフォーマンス監視

### 3.5.2 サポート体制

- ユーザーサポート窓口
- FAQ・ヘルプドキュメント
- チュートリアル・ガイド
- 問い合わせ対応システム

# 4. 技術選定

## 4.1 技術スタック選定の方針

### 4.1.1 選定基準

- 開発効率性：迅速な開発とメンテナンスが可能
- スケーラビリティ：将来的な機能拡張に対応可能
- セキュリティ：個人情報を扱うため高いセキュリティレベル
- コミュニティサポート：豊富なドキュメントと活発なコミュニティ
- 運用コスト：適切なコストでの運用が可能

### 4.1.2 Stack Overflow Developer Survey 2024の調査結果参考

2024年のStack Overflow Developer Surveyによると、以下の技術が高い人気を誇っています：

**プログラミング言語（使用率）：**

- JavaScript: 62.3%
- HTML/CSS: 52.9%
- Python: 51%
- SQL: 51%
- TypeScript: 38.5%

**データベース（使用率）：**

- PostgreSQL: 48.7%
- MySQL: 40.3%
- SQLite: 33.1%

- Microsoft SQL Server: 25.3%
- MongoDB: 24.8%

## 4.2 推奨技術スタック

### 4.2.1 フロントエンド

**React.js + TypeScript**

- 理由:
  - 既存のホームページがReactで構築済み
  - TypeScriptによる型安全性でバグの削減
  - 豊富なライブラリとコンポーネント
  - 高いパフォーマンスと保守性

**UI/UXライブラリ:**

- Tailwind CSS：レスポンシブデザインの効率的な実装
- Shadcn/ui：高品質なUIコンポーネント
- Lucide React：アイコンライブラリ

### 4.2.2 バックエンド

**Node.js + Express.js**

- 理由:
  - JavaScriptの統一による開発効率向上
  - 豊富なnpmパッケージエコシステム
  - 非同期処理に優れた性能
  - RESTful APIの構築が容易

**代替案：Python + Flask/FastAPI**

- 理由:
  - データ処理・分析に優れた性能
  - 機械学習ライブラリとの親和性
  - シンプルで読みやすいコード

### 4.2.3 データベース

**PostgreSQL（メインデータベース）**

- 理由:
  - 最も人気の高いデータベース（48.7%使用率）
  - ACID準拠による高い信頼性
  - 複雑なクエリとJSONデータ型のサポート
  - 優れたパフォーマンスとスケーラビリティ

**Redis（キャッシュ・セッション管理）**

- 理由:
  - 高速なインメモリデータストア
  - セッション管理とキャッシュに最適
  - リアルタイム機能のサポート

### 4.2.4 認証・認可

**JSON Web Token (JWT) + bcrypt**

- 理由:
  - ステートレスな認証システム
  - スケーラブルな設計
  - 業界標準のセキュリティ実装

**OAuth 2.0対応**

- Google、Facebook等のソーシャルログイン
- 高齢者にも使いやすい認証オプション

### 4.2.5 ファイルストレージ

**AWS S3 / Cloudflare R2**

- 理由:
  - 写真・動画・音声ファイルの安全な保存
  - CDNによる高速配信
  - 自動バックアップとバージョン管理

### 4.2.6 デプロイメント・インフラ

**Docker + Kubernetes / Vercel + Railway**

- 理由:
  - コンテナ化による環境の一貫性

- 自動スケーリングと負荷分散
- CI/CDパイプラインの構築

## 4.3 開発環境・ツール

### 4.3.1 開発ツール

- **IDE：** Visual Studio Code
- **バージョン管理：** Git + GitHub
- **パッケージマネージャー：** pnpm（Node.js）、pip（Python）
- **API開発・テスト：** Postman / Thunder Client

### 4.3.2 品質管理

- **コード品質：** ESLint、Prettier
- **テスト：** Jest（単体テスト）、Cypress（E2Eテスト）
- **型チェック：** TypeScript
- **セキュリティ：** npm audit、Snyk

### 4.3.3 監視・ログ

- **アプリケーション監視：** Sentry
- **ログ管理：** Winston（Node.js）
- **パフォーマンス監視：** New Relic / DataDog

## 4.4 セキュリティ対策

### 4.4.1 データ保護

- **暗号化：** AES-256による保存時暗号化
- **通信：** TLS 1.3による通信暗号化
- **パスワード：** bcryptによるハッシュ化（ソルト付き）

### 4.4.2 アクセス制御

- **認証：** JWT + リフレッシュトークン
- **認可：** RBAC（Role-Based Access Control）
- **API保護：** レート制限、CORS設定

#### 4.4.3 脆弱性対策

- **入力検証：** バリデーションライブラリの使用
- **SQLインジェクション：** パラメータ化クエリ
- **XSS対策：** CSP（Content Security Policy）設定

# 5. 開発計画

## 5.1 開発フェーズ

### フェーズ1：基盤構築（2-3週間）

- データベース設計・構築
- 基本的なAPI設計
- 認証システムの実装
- 基本的なフロントエンド構造

### フェーズ2：コア機能開発（4-6週間）

- ユーザー管理機能
- デジタル墓碑・追悼ページ機能
- ファイルアップロード機能
- 基本的な公開範囲設定

### フェーズ3：拡張機能開発（3-4週間）

- バーチャルお参り機能
- デジタル家系図機能
- ライフログ・未来への手紙機能
- コミュニティ機能

### フェーズ4：統合・テスト（2-3週間）

- 機能統合とテスト
- セキュリティテスト
- パフォーマンステスト
- ユーザビリティテスト

#### フェーズ5：デプロイ・運用（1-2週間）

- 本番環境構築
- デプロイメント
- 監視システム設定
- ドキュメント整備

### 5.2 リスク管理

#### 5.2.1 技術的リスク

- **データベース設計の複雑性：** 段階的な設計とプロトタイプ検証
- **ファイルストレージの容量：** クラウドストレージの適切な選択
- **パフォーマンス問題：** 早期のパフォーマンステスト実施

#### 5.2.2 ユーザビリティリスク

- **高齢者の操作性：** ユーザーテストの実施
- **アクセシビリティ：** WCAG 2.1準拠の実装
- **多言語対応：** 国際化対応の設計

#### 5.2.3 セキュリティリスク

- **個人情報漏洩：** 定期的なセキュリティ監査
- **不正アクセス：** 多層防御の実装
- **データ消失：** 自動バックアップシステム

### 5.3 成功指標（KPI）

#### 5.3.1 技術指標

- **応答時間：** 平均3秒以内
- **稼働率：** 99.9%以上
- **セキュリティ：** 脆弱性ゼロ
- **パフォーマンス：** Lighthouse スコア90以上

#### 5.3.2 ユーザー指標

- **ユーザー登録数：** 初月100人以上

- **アクティブユーザー率：** 60%以上
- **ユーザー満足度：** 4.5/5.0以上
- **機能利用率：** 各機能50%以上

# 6. まとめ

## 6.1 要件定義の概要

本要件定義書では、Nexia（ネクシア）ウェブアプリケーションの開発に必要な機能要件、非機能要件、技術選定、開発計画を詳細に定義しました。

## 6.2 重要なポイント

1. **ユーザー中心設計：** 高齢者にも使いやすいインターフェース
2. **セキュリティ重視：** 個人情報保護の徹底
3. **スケーラビリティ：** 将来的な機能拡張への対応
4. **技術的安定性：** 実績のある技術スタックの採用

## 6.3 次のステップ

1. データベース設計の詳細化
2. API仕様書の作成
3. UI/UXデザインの作成
4. 開発環境の構築
5. 開発チームの編成

この要件定義書を基に、段階的にウェブアプリケーションの開発を進めていきます。各フェーズでユーザーフィードバックを収集し、継続的な改善を行うことで、高品質なサービスの提供を目指します。

---

**作成者：** Manus AI

**作成日：** 2025年8月16日

**バージョン：** 1.0