Scope Checklist for COMP1682 - Final Year Project

Author: Vinh Hoang

Date: 06 Jul

Version: 0.1

# Full Stack Web Application

* Develop a feature-rich web application with both frontend and backend components.
  + Utilize a modern frontend framework/library such as React, Angular, or Vue.js.
  + Implement responsive design for seamless user experience across devices.
  + Use a backend framework like Node.js, Django, or Ruby on Rails, .NET
  + Incorporate a database system like MySQL, PostgreSQL, or MongoDB.
* Integrate a third-party payment API (e.g., Stripe, PayPal) for secure online transactions.
* Include social media integration (e.g., Facebook, Twitter, Google) for user authentication and sharing.
* Implement real-time updates using a WebSocket library like Socket.io or Pusher.
* Utilize a cloud storage service (e.g., AWS S3, Google Cloud Storage, Cloudinary) for file uploads and downloads.
* Integrate a geolocation API (e.g., Google Maps) for location-based features.
* The more services, the better
* Implement a caching mechanism (e.g., Redis) for improved performance.
* Deployed on real host such as Netlify, Render, AWS, Azure, etc.

# Front-End Focus

* Develop a visually appealing and interactive user interface.
* Utilize modern frontend frameworks/libraries like React, Angular, or Vue.js.
* Implement responsive design for optimal viewing on different devices.
* Integrate third-party APIs for features like payment processing, real-time chat, social media login, or authentication such as (Google, Facebook, Firebase, Stripe, Paypal) **(the more services the better)**
* Use Back-end-as-a-Service like Firebase
* Use a UI component library (e.g., Material-UI, Ant Design) for efficient development.
* Implement state management using Redux, Vuex, or MobX.
* Utilize a styling solution like CSS preprocessors (e.g., Sass, Less) or CSS-in-JS (e.g., styled-components).
* Implement performance optimization techniques (e.g., lazy loading, code splitting) for faster page load times.
* Incorporate accessibility features to ensure inclusive user experience.
* Integrate analytics tools (e.g., Google Analytics) for tracking user behavior and performance metrics.

# Back-End Focus

* Develop a robust and scalable backend system with Microservices Architecture or Serverless Architecture
* Use a backend framework like Node.js, Django, Ruby on Rails, or ASP.NET.
* Implement database management and modeling using systems like MySQL, PostgreSQL, or MongoDB.
* Implement an API architecture (e.g., RESTful API, GraphQL) for communication with frontend or mobile applications.
* Integrate third-party APIs for features like payment processing, SMS notifications, email services, etc. **(the more services the better)**
* Implement authentication and authorization mechanisms (e.g., JWT, OAuth) for secure access control.
* Use caching mechanisms (e.g., Redis, Memcached) for performance optimization.
* Implement background processing or task scheduling (e.g., Celery, Sidekiq) for asynchronous operations.
* Utilize logging and error tracking tools (e.g., Sentry, ELK Stack) for monitoring and debugging purposes.
* Implement security measures like input validation, encryption, and sanitization to prevent vulnerabilities.
* Integrate DevOps, CI/CD pipeline

# Mobile Application Focus

* Develop a native or hybrid mobile application for iOS or Android platforms.
* Use frameworks/libraries like React Native, Flutter, or Xamarin for cross-platform development.
* Implement a user-friendly and intuitive user interface following platform-specific design guidelines.
* Integrate third-party APIs for features like payment processing, push notifications, social media login, etc. **(the more services the better)**
* Implement offline data synchronization for seamless user experience in limited or no network connectivity.
* Utilize device-specific features like camera, GPS, accelerometer, etc., for enhanced functionality.
* Implement secure authentication and data encryption for user privacy.
* Integrate analytics tools (e.g., Firebase Analytics, Google Analytics) for tracking user behavior and app performance.
* Incorporate in-app purchase functionality for monetization of the application (if applicable).
* Implement performance optimization techniques (e.g., lazy loading, code splitting) for faster app loading and smooth user experience.
* Use testing frameworks (e.g., Jest, XCTest) to ensure the quality and reliability of the application.
* Implement crash reporting and error tracking tools (e.g., Crashlytics, Sentry) for monitoring and debugging purposes.