# 

TeckedIn Backend Application Design Specification

|  |  |  |
| --- | --- | --- |
| **Author** | **Revision Number** | **Date** |
| Architect | 1.0 |  |
|  |  |  |
|  |  |  |

[TeckedIn Backend Application Design Specification 1](#_Toc489836068)

[Application Design Specification 3](#_Toc489836069)

[1. Design 3](#_Toc489836070)

[1.1 Work Flow Description 3](#_Toc489836071)

[1.2 Component Requirements 3](#_Toc489836072)

[1.2.1 Custom Software Components 3](#_Toc489836073)

[1.2.2 TopCoder Generic Components 3](#_Toc489836074)

[1.2.3 Third Party Components 3](#_Toc489836075)

[1.2.4 Assemblies 3](#_Toc489836076)

[1.3 Application Management 3](#_Toc489836077)

[1.3.1 Transaction 3](#_Toc489836078)

[1.3.2 Configuration 3](#_Toc489836079)

[1.3.3 Persistence 4](#_Toc489836080)

[1.3.4 Thread-Safety and Concurrency 4](#_Toc489836081)

[1.3.5 Logging 4](#_Toc489836082)

[1.3.6 Auditing 4](#_Toc489836083)

[1.3.7 Exception Handling 4](#_Toc489836084)

[1.3.8 Internationalization 4](#_Toc489836085)

[1.3.9 Security 4](#_Toc489836086)

[1.3.10 Performance 4](#_Toc489836087)

[1.3.11 Scalability 4](#_Toc489836088)

[1.4 Deployment Constraints 4](#_Toc489836089)

[1.4.1 Technology overview 5](#_Toc489836090)

[1.5 Development Standards: 5](#_Toc489836091)

[1.6 Interfaces Classes Overview 5](#_Toc489836092)

[1.7 Changes to Existing System 5](#_Toc489836093)

[2. User Interface 5](#_Toc489836094)

[3. Included Documentation 5](#_Toc489836095)

[3.1 Architecture Documentation 5](#_Toc489836096)

[4. Future Enhancements 5](#_Toc489836097)

# Application Design Specification

# Design

The goal of this project is to build a TeckedIn portal that connects technology providers with technology users.

## Work Flow Description

The portal will consist of frontend and backend.

Internally backend implements a set of Node.js modules as well as Express controllers to expose REST APIs.

The backend will access MongoDB database via Mongoose library.

## Component Requirements

### Custom Software Components

#### *New Custom Components*

None

### TopCoder Generic Components

None

### Third Party Components

* [Node.js 8](http://nodejs.org)

This application is built on Node.js platform.

* [Express 4.15](http://expressjs.com)

This is used as the web framework.

* [MongoDB 3.4](http://www.mongodb.org)

This is the database system used in this application.

* [Mongoose.js 4.11](https://www.npmjs.org/package/mongoose)

This is used to simplify the interaction with MongoDB.

* [winston 2.3](https://www.npmjs.org/package/winston)

This is used for logging.

* [passport 0.3.2](https://www.npmjs.org/package/passport)

This is used to support authentication for the REST APIs.

### Assemblies

* TeckedIn Backend REST APIs Assembly

This assembly will implement the backend of the application.

## Application Management

### Transaction

MongoDB doesn't support complex transaction management like conventional RDBMS, but it does support [atomic operations on single documents](http://www.mongodb.org/display/DOCS/Atomic+Operations), which can guarantee atomic operations on single nested objects (aka "documents" in MongoDB terminology), and that's sufficient for this project.

### Configuration

The backend configurations will be specified in JavaScript file config.js, all configurations are clearly shown on "Backend Class Diagram" and all configuration options are required.

JavaScript services and main application script will import the config.js as a Node.js module using "require" statement.

### Persistence

Application data are stored in the [MongoDB](http://www.mongodb.org), and [mongoose](http://mongoosejs.com) is used as the ODM library for easy MongoDB access.

Data schemas are defined on "Models Class Diagram".

### Thread-Safety and Concurrency

Applications built with JavaScript and Node.js is inherently asynchronous, callbacks(JavaScript) are used extensively to coordinate asynchronous tasks/operations. There's generally no thread safety concern with Node.js powered applications.

### Logging

The backend JavaScript code will perform logging using [winston](https://www.npmjs.org/package/winston) Node.js module.

Errors will be logged at ERROR level, debug information will be logged at DEBUG level.

User/client and database credentials must not be logged.

### Auditing

None

### Exception Handling

The functions of JavaScript services are implemented in asynchronous manner, i.e. virtually all functions will take a callback function that will be called to notify function caller of result. If error occurs, callback function will be called with an Error type parameter "error" detailing the error.

Node.js/Express controllers will interpret errors as HTTP status codes

* 400 The request could not be interpreted correctly or some required parameters were missing.
* 404 The entity to update does not exist.
* 401 The request didn't include authentication information.
* 405 The requested method is not supported.
* 403 The request was forbidden because of insufficient permission
* 500 Something is broken at API server-side.

### Internationalization

There is no internationalization requirement.

### Security

The backend REST APIs requires bearer authentication.

[passport](https://www.npmjs.com/package/passport) will be used to implement the authentication. Once authenticated, a token will be issued to REST API client, so that the user identity can be extracted in subsequent API requests until logout or expiration.

The REST APIs should be exposed through HTTPS.

### Performance

Mongoose internally uses a connection pool to improve performance, therefore it is not necessary for this application to manage connection pool by itself.

Pagination is supported for the search REST APIs to improve query experience.

### Scalability

There is no particular scalability requirement, and the proposed architecture does not prevent the application from being scalable.

## Deployment Constraints

There is no particular scalability requirement.

* REST API will expose REST API endpoints to frontend.

It will run as a Node.js/Express application.

### Technology overview

* JavaScript
* JSON
* REST
* SMTP
* Backend
  + Node.js 8
  + Express 4.15
  + MongoDB 3.4
  + Mongoose.js 4.11
  + winston 2.3
  + passport 0.3.2

## Development Standards:

The assembly development must adhere to the guidelines as outlined in the [TopCoder Assembly Competition Tutorial](http://apps.topcoder.com/wiki/display/tc/Assembly+Competition+Tutorials).

## Interfaces Classes Overview

See the TCUML file.

## Changes to Existing System

None

# User Interface

The frontend will be built in HTML5 and AngularJS and will consume backend REST APIs as per guidelines provided in Integration Assembly Specification.

# Included Documentation

## Architecture Documentation

* Class Diagrams
* Sequence Diagrams
* Application Design Specification
* Assembly Specifications
* REST API Specification

# Future Enhancements

None