# **ID607001: Introductory Application Development Concepts**

# **Project Marking Rubric**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **10-9** | **8-7** | **6-5** | **4-0** |
| **Functionality – Your choice REST API** | The Express REST API developed using Node.js contains comprehensive and robust evidence on the following functionality: development and production modification, models, data types, relationships, enum, files, messages, filtering, sorting, pagination, a 404 endpoint, validation, Swagger documentation, PostgreSQL database, deployment and scripts. | The Express REST API developed using Node.js contains clear and detailed evidence on the following functionality: development and production modification, models, data types, relationships, enum, files, messages, filtering, sorting, pagination, a 404 endpoint, validation, Swagger documentation, PostgreSQL database, deployment and scripts. | The Express REST API developed using Node.js contains evidence on the following functionality: development and production modification, models, data types, relationships, enum, files, messages, filtering, sorting, pagination, a 404 endpoint, validation, Swagger documentation, PostgreSQL database, deployment and scripts. | The Express REST API developed using Node.js does not or does not fully contain evidence on the following functionality: development and production modification, models, data types, relationships, enum, files, messages, filtering, sorting, pagination, a 404 endpoint, validation, Swagger documentation, PostgreSQL database, deployment and scripts. |
|  | **10-9** | **8-7** | **6-5** | **4-0** |
| **Functionality – OpenTDB API** | The Express REST API developed using Node.js contains comprehensive and robust evidence on the following functionality: development and production modification, enums, models, admin user, basic user, validation, seeding, Helmet, CORS, rate limiting, compression, Swagger documentation, PostgreSQL database, deployment and scripts. | The Express REST API developed using Node.js contains clear and detailed evidence on the following functionality: development and production modification, enums, models, admin user, basic user, validation, seeding, Helmet, CORS, rate limiting, compression, Swagger documentation, PostgreSQL database, deployment and scripts. | The Express REST API developed using Node.js contains evidence on the following functionality: development and production modification, enums, models, admin user, basic user, validation, seeding, Helmet, CORS, rate limiting, compression, Swagger documentation, PostgreSQL database, deployment and scripts. | The Express REST API developed using Node.js does not or does not fully contain evidence on the following functionality: development and production modification, enums, models, admin user, basic user, validation, seeding, Helmet, CORS, rate limiting, compression, Swagger documentation, PostgreSQL database, deployment and scripts. |
| **Code Quality and Best Practices** | The Express REST API developed using Node.js demonstrate comprehensive evidence on the following:   * Environment variables. * Appropriate naming. * Idiomatic use. * Efficient algorithmic approach. * Sufficient modularity. * JSDoc header comment. * Code is formatted. * No dead or unused code. | The Express REST API developed using Node.js demonstrate clear evidence on the following:   * Environment variables. * Appropriate naming. * Idiomatic use. * Efficient algorithmic approach. * Sufficient modularity. * JSDoc header comment. * Code is formatted. * No dead or unused code. | The Express REST API developed using Node.js demonstrate evidence on the following:   * Environment variables. * Appropriate naming. * Idiomatic use. * Efficient algorithmic approach. * Sufficient modularity. * JSDoc header comment. * Code is formatted. * No dead or unused code. | The Express REST API developed using Node.js do not or do not fully demonstrate evidence on the following:   * Environment variables. * Appropriate naming. * Idiomatic use. * Efficient algorithmic approach. * Sufficient modularity. * JSDoc header comment. * Code is formatted. * No dead or unused code. |
| **Documentation and Git Usage** | Comprehensive use of project board or issues on GitHub.  README file contains comprehensive evidence on the following:   * A URL to your REST APIs as web service on Render. * Setup the environment. * Run your REST APIs locally. * Create and apply a migration. * Reset the PostgreSQL database. * Seed users. * Open Prisma Studio. * Check your code. * Format your code. * An ERD of your REST APIs. * Use of Markdown. * Spelling and grammar correctness.   Git commit messages are comprehensively formatted and reflect the changes in concise detail. | Clear use of project board or issues on GitHub.  README file contains clear evidence of:   * A URL to your REST APIs as web service on Render. * Setup the environment. * Run your REST APIs locally. * Create and apply a migration. * Reset the PostgreSQL database. * Seed users. * Open Prisma Studio. * Check your code. * Format your code. * An ERD of your REST APIs. * Use of Markdown. * Spelling and grammar correctness.   Git commit messages are clearly formatted and reflect the changes in substantial detail. | Use of project board or issues on GitHub.  README file contains evidence of:   * A URL to your REST APIs as web service on Render. * Setup the environment. * Run your REST APIs locally. * Create and apply a migration. * Reset the PostgreSQL database. * Seed users. * Open Prisma Studio. * Check your code. * Format your code. * An ERD of your REST APIs. * Use of Markdown. * Spelling and grammar correctness.   Git commit messages are formatted and reflect the changes in detail. | Does not or does not full demonstrate use of project board or issues on GitHub.  README file does not or does not fully contain evidence of:   * A URL to your REST APIs as web service on Render. * Setup the environment. * Run your REST APIs locally. * Create and apply a migration. * Reset the PostgreSQL database. * Seed users. * Open Prisma Studio. * Check your code. * Format your code. * An ERD of your REST APIs. * Use of Markdown. * Spelling and grammar correctness.   Git commit messages are not or are not fully formatted and do not or do not fully reflect the changes. |

# **ID607001: Introductory Application Development Concepts**

# **Project Marking Cover Sheet**

Name:

Date:

Learner ID:

Assessor’s Name:

Assessor’s Signature:

|  |  |  |  |
| --- | --- | --- | --- |
| **Criteria** | **Out Of** | **Weighting** | **Final Result** |
| Functionality | 10 | 50 |  |
| Code Quality and Best Practices | 10 | 40 |  |
| Documentation and Git Usage | 10 | 10 |  |
| **Final Result** | | | /100 |
| **This assessment is worth 80% of the final mark for the Introductory Application Development Concepts course.** | | | |

**Feedback:**

**Functionality:**

**Code Quality and Best Practices:**

**Documentation and Git Usage:**