Introduction:

The project's backend application makes use of Node.js, Express.js, and MongoDB to deliver a reliable and scalable solution. REST APIs for managing favorites, leaders, promotions, authentication, file uploads, and dishes are just a few of its standout features. Passport.js, JWT, and OAuth2 with Facebook are also used in the application as secure authentication methods. Additionally, implementing SSL certificates enables safe connection, and addressing CORS improves compatibility.

Design and Implementation:

System Description:

Through REST APIs, the backend application is built to handle a variety of functionalities. These comprise:

Manages actions involving dishes, including retrieving, creating, updating, and deleting dishes.

User authentication API: Offers safe user login and registration features.

Users can upload files to the server using the upload file API, which simplifies this capability.

Using the Favorite Dishes API, users may save and get their favorite dishes.

Design and Implementation Details:

The project underwent a smooth transition from the early design stage to actual realization thanks to a carefully thought-out implementation procedure. According to the design guidelines, the code was implemented using Node.js and Express.js. To address problems that arose during implementation and boost the system's overall efficacy and efficiency, some changes were implemented.

Modules and Libraries:

In order to implement the backend application, a number of modules and libraries were used. Important elements consist of

body-parser: A middleware module used to parse incoming request bodies in a readable format.

cors: A module that enables Cross-Origin Resource Sharing, allowing requests from different domains.

express: The core framework used for building the backend application, providing routing, middleware, and handling HTTP requests.

express-session: A module that enables session management and storage for Express.js applications.

jsonwebtoken: A library for generating and verifying JSON Web Tokens (JWTs) used in authentication and authorization processes.

mongoose: A MongoDB object modeling library for Node.js, providing a convenient way to interact with the database.

mongoose-currency: A module that adds support for handling currency fields in Mongoose schemas.

multer: A middleware for handling file uploads, providing support for multipart/form-data requests.

passport: A module that simplifies authentication processes by providing an extensible framework with various authentication strategies.

passport-facebook-token: A Passport.js strategy for authenticating users using Facebook access tokens.

passport-jwt: A Passport.js strategy for authenticating users using JSON Web Tokens (JWTs).

passport-local: A Passport.js strategy for authenticating users with username and password credentials.

passport-local-mongoose: A Mongoose plugin that simplifies building username and password authentication with Passport.js.

Conclusions:

Results Obtained:

The backend application successfully delivered the intended functionality, allowing users to interact with various APIs and perform operations on dishes, leaders, promotions, authentication, file uploads, and favorite dishes. The implemented authentication mechanisms ensured secure user access, while SSL certificate implementation enhanced data protection during communication. CORS handling facilitated seamless interaction with the application from different domains.

Features and Shortcomings:

The project showcased several notable features, including a comprehensive set of REST APIs, robust authentication mechanisms, and secure communication through SSL certificates. However, certain areas for improvement and potential shortcomings were identified, such as [mention any identified shortcomings or areas for further development].

Choices in Hindsight:

Upon completing the project, certain choices could have been approached differently in hindsight. For instance, [discuss specific decisions that, with hindsight, could have been altered]. Reflecting on these choices provides valuable insights for future projects, facilitating continuous improvement.

References:

Node.js: https://nodejs.org/

Express.js: https://expressjs.com/

MongoDB: https://www.mongodb.com/

Passport.js: http://www.passportjs.org/

JSON Web Token (JWT): https://jwt.io/

OAuth2: https://oauth.net/2/

Facebook for Developers: https://developers.facebook.com/

CORS: https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS