SWDV691 – CAPSTONE

DESIGN – SERVICE

Submitted by: Mia Jacobs

OVERVIEW:

Tech Talk will utilize NodeJS JavaScript runtime on the backend of the application. This is a server-side component that is responsible for handling the logic of the application. It is built using the Node.js runtime environment, which will allow for the ability to write server-side code using JavaScript, the same language used on the client side.

The Tech Talk backend will interact with the database, handle authentication, and manage user data. It provides APIs that client-side components can use to communicate with the server and retrieve or update data. The Node.js backend will be built using various packages and frameworks. These will help create a robust and scalable backend that can handle high traffic, data processing, and storage.

The application backend has the potential to integrate with various social media platforms to allow users to log in and share content, such as posts, photos, and videos. It also can use machine learning algorithms to provide personalized recommendations to users and detect fraud or spam. Overall, this Node.js backend is a crucial part of the application architecture and plays a vital role in providing a seamless user experience.

LAYERS:

1. Presentation Layer: This layer will be responsible for handling the user interface and user interactions. It will provide a platform for users to interact with the application through web or mobile devices.
2. Application Layer: This layer will handle the logic of the application. It is responsible for creating and maintaining the state of the application, processing input from the presentation layer, and generating output for the presentation layer.
3. Data Access Layer: This layer will provide an abstraction for accessing the data in the database. It manages the connections to the database and executes queries. It also handles any data transformation and validation before it is sent to the application layer.
4. Database Layer: This layer will handle the storage and retrieval of data. It is responsible for creating and maintaining the database schema, managing data transactions, and enforcing data integrity.
5. Infrastructure Layer: This layer will provide the foundation for the application to run. It includes the operating system, web server, network infrastructure, and other software that is required to run the application.

SERVICES THAT INTERACT WITH THE DATABASE:

1. API (Application Programming Interface): An API is a set of protocols and tools for building software applications that enable the communication between applications and databases. The social networking application can provide an API that UI designers can use to interact with the database.
2. Database Management System: A database management system is software that manages data and allows users to interact with the data. UI designers can use a database management system to manage and interact with the social networking application database.
3. Backend Development Frameworks: Backend development frameworks like Django, Ruby on Rails, or Laravel can help developers to turn database designs into functional products. UI designers can use backend development frameworks to create the design for the social networking application database and interact with it.
4. Cloud-Based Storage Services: Amazon AWS will be used to store and manage social networking application data. UI designers can design and develop the user interface to interact with cloud-based storage services.
5. Frontend Development Frameworks: Frontend React will be used to develop UIs that can interact with social networking application databases. UI designers can use front-end development frameworks to develop a user interface that interacts with the database.

SERVICE ENDPOINTS:

1. User Registration and Login endpoint: This endpoint allows users to create new accounts and sign into their accounts.

A picture containing icon

Description automatically generated

|  |  |  |
| --- | --- | --- |
| Login REQUEST:  POST /api/login HTTP/1.1  Host: example.com  Content-Type: application/json  {  "email": "user@example.com",  "password": "secretpassword"  } |  | RESPONSE:  HTTP/1.1 200 OK  Content-Type: application/json  {  "user\_id": 1234,  "access\_token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",  "expires\_in": 3600  } |
| Incorrect Password Error REQUEST:  POST /api/login HTTP/1.1  Host: example.com  Content-Type: application/json  {  "email": "user@example.com",  "password": "wrongpassword"  } |  | RESPONSE:  HTTP/1.1 401 Unauthorized  Content-Type: application/json  {  "error": "Incorrect password"  } |
| User not found error REQUEST:  POST /api/login HTTP/1.1  Host: example.com  Content-Type: application/json  {  "email": "nonexistentuser@example.com",  "password": "password"  } |  | RESPONSE:  HTTP/1.1 404 Not Found  Content-Type: application/json  {  "error": "User not found"  } |
| Invalid Request Error REQUEST:  POST /api/login HTTP/1.1  Host: example.com  Content-Type: application/json  {  "username": "user@example.com",  "password": "secretpassword"  } |  | RESPONSE:  HTTP/1.1 400 Bad Request  Content-Type: application/json  {  "error": "Invalid request"  } |

1. User Profile endpoint: This endpoint will let users update their profile information, including profile picture, bio, education, work experience, and contacts.

A picture containing icon

Description automatically generated

|  |  |  |
| --- | --- | --- |
| REQUEST:  GET /api/profile HTTP/1.1  Host: example.com  Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9... |  | RESPONSE:  HTTP/1.1 200 OK  Content-Type: application/json  {  "user\_id": 1234,  "name": "John Doe",  "email": "johndoe@example.com",  "profile\_picture": "https://example.com/profile.jpg",  "bio": "I am a software engineer"  } |
| Unauthorized Error REQUEST:  GET /api/profile HTTP/1.1  Host: example.com  Authorization: Bearer invalidtoken |  | RESPONSE:  HTTP/1.1 401 Unauthorized  Content-Type: application/json  {  "error": "Unauthorized"  } |
| User not Found Error REQUEST:  GET /api/profile/9999 HTTP/1.1  Host: example.com  Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9... |  | RESPONSE:  HTTP/1.1 404 Not Found  Content-Type: application/json  {  "error": "User not found"  } |
| Invalid Request Error REQUEST:  GET /api/profile HTTP/1.1  Host: example.com  Authorization: Basic dXNlcjpwYXNzd29yZA== |  | RESPONSE:  HTTP/1.1 400 Bad Request  Content-Type: application/json  {  "error": "Invalid request"  } |

1. Search endpoint: This endpoint will allow users to search for other users' profiles, posts, and work experience based on different criteria, such as keywords, location, industry, etc.

A picture containing icon

Description automatically generated

|  |  |  |
| --- | --- | --- |
| Successful Search REQUEST:  GET /api/search?q=john HTTP/1.1  Host: example.com  Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9... |  | RESPONSE:  HTTP/1.1 200 OK  Content-Type: application/json  {  "results": [  {  "user\_id": 1234,  "name": "John Doe",  "profile\_picture": "https://example.com/profile.jpg"  },  {  "user\_id": 5678,  "name": "Jane Johnson",  "profile\_picture": "https://example.com/profile.jpg"  }  ]  } |
| Unauthorized Error REQUEST:  GET /api/search?q=john HTTP/1.1  Host: example.com  Authorization: Basic dXNlcjpwYXNzd29yZA== |  | RESPONSE:  HTTP/1.1 401 Unauthorized  Content-Type: application/json  {  "error": "Unauthorized"  } |
| Invalid Query Error REQUEST:  GET /api/search?q= HTTP/1.1  Host: example.com  Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9... |  | RESPONSE:  HTTP/1.1 400 Bad Request  Content-Type: application/json  {  "error": "Invalid query"  } |
| Server Error REQUEST:  GET /api/search?q=john HTTP/1.1  Host: example.com  Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9... |  | RESPONSE:  HTTP/1.1 500 Internal Server Error  Content-Type: application/json  {  "error": "Internal server error"  } |

1. Friends endpoint: To search and connect with other users on the application. It allows users to view each other's profiles and interact privately via messaging.

A picture containing icon

Description automatically generated

|  |  |  |
| --- | --- | --- |
| Successful REQUEST:  GET /api/friends |  | RESPONSE:  Status: 200 OK  Content-Type: application/json  [  {  "id": 1,  "name": "John",  "age": 28  },  {  "id": 2,  "name": "Jane",  "age": 35  },  {  "id": 3,  "name": "Bob",  "age": 22  }  ] |
| REQUEST:  POST /api/friends  Content-Type: application/json  {  "name": "Alice",  "age": 30  } |  | RESPONSE:  Status: 201 Created  Content-Type: application/json  {  "id": 4,  "name": "Alice",  "age": 30  } |
| REQUEST:  PUT /api/friends/2  Content-Type: application/json  {  "name": "Janet",  "age": 40  } |  | RESPONSE:  Status: 200 OK  Content-Type: application/json  {  "id": 2,  "name": "Janet",  "age": 40  } |
| REQUEST:  DELETE /api/friends/3 |  | RESPONSE:  Status: 204 No Content |
| Erroneous REQUEST:  POST /api/friends  Content-Type: application/json  {  "age": 25  } |  | RESPONSE:  Status: 400 Bad Request  Content-Type: application/json  {  "error": "Name is required"  } |
| Erroneous REQUEST:  PUT /api/friends/5  Content-Type: application/json  {  "name": "Sam"  } |  | RESPONSE:  Status: 400 Bad Request  Content-Type: application/json  {  "error": "Age is required"  } |

1. Post endpoint: To enable users to create and share content like posts, images, videos, and links. This endpoint also allows users to view, like, and comment on other users' posts.

A picture containing icon

Description automatically generated

|  |  |  |
| --- | --- | --- |
| Successful REQUEST:  GET /api/posts |  | RESPONSE:  Status: 200 OK  Content-Type: application/json  [  {  "id": 1,  "title": "First Post",  "content": "This is my first post on this site."  },  {  "id": 2,  "title": "Second Post",  "content": "This is my second post on this site."  },  {  "id": 3,  "title": "Third Post",  "content": "This is my third post on this site."  }  ] |
| Successful REQUEST:  POST /api/posts  Content-Type: application/json  {  "title": "Fourth Post",  "content": "This is my fourth post on this site."  } |  | RESPONSE:  Status: 201 Created  Content-Type: application/json  {  "id": 4,  "title": "Fourth Post",  "content": "This is my fourth post on this site."  } |
| Successful REQUEST:  PUT /api/posts/2  Content-Type: application/json  {  "title": "Updated Second Post",  "content": "This is an updated version of my second post."  } |  | RESPONSE:  Status: 200 OK  Content-Type: application/json  {  "id": 2,  "title": "Updated Second Post",  "content": "This is an updated version of my second post."  } |
| Successful REQUEST:  DELETE /api/posts/3 |  | RESPONSE:  Status: 204 No Content |
| Erroneous REQUEST  POST /api/posts  Content-Type: application/json  {  “content”: “This is a post without a title.”  } |  | RESPONSE:  Status: 400 Bad Request  Content-Type: application/json  {  "error": "Title is required"  } |
| Erroneous REQUEST:  PUT /api/posts/5  Content-Type: application/json  {  "title": "Updated Post"  } |  | RESPONSE:  Status: 400 Bad Request  Content-Type: application/json  {  "error": "Content is required"  } |

1. Messaging endpoint: This endpoint allows users to send private messages to each other.

A picture containing icon

Description automatically generated

|  |  |  |
| --- | --- | --- |
| REQUEST:  GET /api/messages |  | RESPONSE:  Status: 200 OK  Content-Type: application/json  [  {  "id": 1,  "sender": "John",  "recipient": "Jane",  "message": "Hello Jane, how are you doing?"  },  {  "id": 2,  "sender": "Jane",  "recipient": "John",  "message": "Hi John, I'm doing well. How about you?"  },  {  "id": 3,  "sender": "Bob",  "recipient": "Alice",  "message": "Hey Alice, did you see the new movie that came out?"  }  ] |
| REQUEST:  POST /api/messages  Content-Type: application/json  {  "sender": "Alice",  "recipient": "Bob",  "message": "Hey Bob, how's it going?"  } |  | RESPONSE:  Status: 201 Created  Content-Type: application/json  {  "id": 4,  "sender": "Alice",  "recipient": "Bob",  "message": "Hey Bob, how's it going?"  } |
| REQUEST:  PUT /api/messages/2  Content-Type: application/json  {  "sender": "Janet",  "recipient": "John",  "message": "Hi John, how are you doing?"  } |  | RESPONSE:  Status: 200 OK  Content-Type: application/json  {  "id": 2,  "sender": "Janet",  "recipient": "John",  "message": "Hi John, how are you doing?"  } |
| REQUEST:  DELETE /api/messages/3 |  | RESPONSE:  Status: 204 No Content |
| Erroneous REQUEST:  POST /api/messages  Content-Type: application/json  {  "sender": "Bob",  "message": "Hey, how's it going?"  } |  | RESPONSE:  Status: 400 Bad Request  Content-Type: application/json  {  "error": "Recipient is required"  } |
| Erroneous REQUEST:  PUT /api/messages/5  Content-Type: application/json  {  "sender": "Alice",  "recipient": "Bob"  } |  | RESPONSE:  Status: 400 Bad Request  Content-Type: application/json  {  "error": "Message is required"  } |

1. Notification endpoint: This endpoint sends push notifications or emails to users to inform them about new messages, likes or comments on their posts, job opportunities, etc.

A picture containing icon

Description automatically generated

|  |  |  |
| --- | --- | --- |
| REQUEST:  GET /api/notifications |  | RESPONSE:  Status: 200 OK  Content-Type: application/json  [  {  "id": 1,  "recipient": "John",  "message": "You have a new friend request from Jane."  },  {  "id": 2,  "recipient": "Jane",  "message": "Bob has accepted your friend request."  },  {  "id": 3,  "recipient": "Bob",  "message": "You have a new message from Alice."  }  ] |
| REQUEST:  POST /api/notifications  Content-Type: application/json  {  "recipient": "Alice",  "message": "You have a new message from Bob."  } |  | RESPONSE:  Status: 201 Created  Content-Type: application/json  {  "id": 4,  "recipient": "Alice",  "message": "You have a new message from Bob."  } |
| REQUEST:  PUT /api/notifications/2  Content-Type: application/json  {  "recipient": "Janet",  "message": "You have a new friend request from John."  } |  | RESPONSE:  Status: 200 OK  Content-Type: application/json  {  "id": 2,  "recipient": "Janet",  "message": "You have a new friend request from John."  } |
| REQUEST:  DELETE /api/notifications/3 |  | RESPONSE:  Status: 204 No Content |
| Erroneous REQUEST:  POST /api/notifications  Content-Type: application/json  {  "message": "You have a new notification."  } |  | RESPONSE:  Status: 400 Bad Request  Content-Type: application/json  {  "error": "Recipient is required"  } |
| Erroneous REQUEST:  PUT /api/notifications/5  Content-Type: application/json  {  "recipient": "Bob"  } |  | RESPONSE:  Status: 400 Bad Request  Content-Type: application/json  {  "error": "Message is required"  } |

1. Search endpoint: To enable users to search for other users or specific content in the application.

A picture containing icon

Description automatically generated

|  |  |  |
| --- | --- | --- |
| GET /search?title=The%20Great%20Gatsby |  | RESPONSE:  HTTP/1.1 200 OK  Content-Type: application/json  {  "results": [  {  "title": "The Great Gatsby",  "author": "F. Scott Fitzgerald",  "publication\_date": "1925",  "available\_copies": 3  }  ]  } |
| Error REQUEST:  GET /search?author= |  | RESPONSE:  HTTP/1.1 400 Bad Request  Content-Type: application/json  {  "error": "Missing author parameter"  } |

1. Comment endpoint: To allow users to comment and express their opinions about posts and other users' comments.

A picture containing icon

Description automatically generated

|  |  |  |
| --- | --- | --- |
| Add Comment:  POST /comments  Content-Type: application/json  {  "post\_id": 123,  "author": "Bob Johnson",  "text": "Nice post. I learned a lot.",  "created\_at": "2021-10-03T09:08:07Z"  } |  | RESPONSE:  HTTP/1.1 201 Created  Content-Type: application/json  {  "id": 3,  "post\_id": 123,  "author": "Bob Johnson",  "text": "Nice post. I learned a lot.",  "created\_at": "2021-10-03T09:08:07Z"  } |
| Update a comment:  PUT /comments/2  Content-Type: application/json  {  "text": "I enjoyed reading this. Keep up the great work!"  } |  | RESPONSE:  HTTP/1.1 200 OK  Content-Type: application/json  {  "id": 2,  "post\_id": 123,  "author": "Jane Smith",  "text": "I enjoyed reading this. Keep up the great work!",  "created\_at": "2021-10-02T10:11:12Z"  } |
| Delete Comment:  DELETE /comments/1 |  | RESPONSE:  HTTP/1.1 204 No Content |
| Erroneous Request – Missing \_id parameter:  GET /comments |  | RESPONSE:  HTTP/1.1 400 Bad Request  Content-Type: application/json  {  "error": "Missing post\_id parameter"  } |
| Errorneous Request – Missing Required Field  POST /comments  Content-Type: application/json  {  "post\_id": 123,  "author": "Bob Johnson"  } |  | RESPONSE:  HTTP/1.1 400 Bad Request  Content-Type: application/json  {  "error": "Missing required field: text"  } |

Overall, each endpoint serves a specific purpose in an IT application, and together, they help users to navigate, connect and network within the platform.

**User Interface pages to Service Endpoints:**

Overall, the communication between the user interface pages and the application's service endpoints is a critical component of social networking applications, as it enables users to interact with one another and the application's features. It is important that this communication is rapid and reliable to ensure a positive user experience.

1. Request Initiation: The user initiates a request by interacting with the user interface page of the social networking application (e.g., sending a message, uploading a photo, or updating a status).
2. Data Transformation: The user's request is then transformed into a format that can be understood by the application's service endpoints, usually in the form of an application programming interface (API). This transformation includes encoding the data in a structured format such as JSON or XML.
3. Data Transmission: The transformed request is then transmitted from the user interface page over the network and to the application's service endpoints.
4. Service Processing: The service endpoints receive the request and process it based on the information contained within the request. For example, the request may be validated, stored, or retrieved from a database.
5. Response Generation: The application's service endpoints then generate a response, which is returned to the user interface page. This response may contain additional data or updates to the user's request.
6. Data Presentation: The user interface page receives the response and presents the data to the user in a format that is easy to understand, such as a notification, a message, or a photo gallery.

**Calls from each page to the backend:**

1. User login and authentication – When a user enters their login credentials and clicks the login button, the frontend application sends an HTTP POST request to the backend server with the user's email and password. The backend server checks the user's credentials against the database and returns a response indicating whether the login was successful or not.

The page should send a request to the backend to authenticate the user with the username and password. The backend then returns an access token that will be used for subsequent requests.

1. User profile – When a user visits their profile page, the frontend application sends an HTTP GET request to the backend server to retrieve the user's profile information, such as their name, profile picture, and bio. If the user is viewing someone else's profile, the frontend application sends an HTTP GET request to the backend server with the user ID of the profile they want to view.

The page must retrieve the user’s profile, which includes their name, profile picture, bio, and other details, from the backend API.

1. Posts – When a user creates a new post, the frontend application sends an HTTP POST request to the backend server with the post content and any associated media, such as photos or videos. The backend server stores the post in the database and returns a response indicating whether the post was successfully created or not.

The page must retrieve the user’s posts, including the images and videos, from the backend API. It may also need to post new content, which requires a request to the backend API to save the post.

1. Feed Page: When a user visits their feed page, the frontend application sends an HTTP GET request to the backend server to retrieve the latest posts from the user's friends and followers. The backend server retrieves the posts from the database and returns them to the frontend application in the response.
2. Comments and likes – The page must retrieve the comments and likes of the user’s posts from the backend API. It may also need to make updates to these interactions, which requires a request to the backend API to store changes.
3. Notifications – The page must retrieve notifications (likes, comments, reactions) from the backend API.
4. Search – When a user searches for other users or content on the platform, the frontend application sends an HTTP GET request to the backend server with the search query. The backend server retrieves the relevant results from the database and returns them to the frontend application in the response.

The page must retrieve search results for users or posts from the backend API.