DOMAIN DRIVEN DESIGN

PROJECT NAME: MASCOT

PROJECT NO: 20

TEAM MEMBERS

TUSHAR (S20210010231)

SREEVALLABH KARANAM (S20210010110)

LAKSHAY YADAV (S20210010129)

VINAY RASTOGI (S20210010240)

VIVEK NEELAPATI (S20210010157)

DOMAIN

SOCIAL MEDIA

SUB DOMAINS

ALUMNI MANAGEMENT

UBIQUITOUS LANGUAGE

VOCABULARY	MEANING
Institute	An educational institution.
Student	Individuals who are studying in a particular institution.
Alumni	Individuals who have graduated from the institution.
Profile	A comprehensive record containing information about an individual, including personal details, contact information, educational history, and professional achievements.
Job Referral	Career opportunities shared with alumni by the employers working with the alumni.
Alumni Network	A community or group of alumni connected through the institution, often facilitating networking, collaboration, and support among its members.
Connection	A user who is the friend of the particular user (can be alumni/student).
Connection Request	A friend request sent to the user. The user may accept or reject the request.

ENTITIES

VOCABULARY	DESCRIPTION	PSEUDO CODE
MongoDocument	The default object structure of the repository. This object will be extended on to every object	`interface IMongoDocument { _id: Schema.Types.ObjectId|string|number __v?: number }`
Profile	A comprehensive record containing information about an individual, including personal details, contact information, educational history, and professional achievements.	`export default interface IProfile extends IMongoDocument { education: (IMongoDocument["_id"]|IEducation)[] achievements: (IMongoDocument["_id"]|IAchievement)[] skills: (IMongoDocument["_id"]|ISkill)[] address: IMongoDocument["_id"]|IAddress workExperience: (IMongoDocument["_id"]|IWork)[] }`
Job Referral	Career opportunities shared with alumni by the employers working with the alumni.	`export default interface IJob extends IMongoDocument { from: IMongoDocument["_id"]|IUser title: string skills: (IMongoDocument["_id"]|ISkill)[] description: string createdAt: Time isCompleted?: boolean endsAt?: Time // The application open time -- need not be set at the time of creation company: (IMongoDocument["_id"]|ICompany) experienceYears: number }`
Connection	A user who is the friend of the particular user (can be alumni/student).	`export default interface IConnection extends IMongoDocument { users: (IMongoDocument["_id"]|IUser)[] }`
Connection Request	A friend request sent to the user. The user may accept or reject the request.	`export default interface IConnectionRequest extends IMongoDocument { from: IMongoDocument["_id"]|IUser to: IMongoDocument["_id"]|IUser type: ConnectionTypes|string createdAt: Time document?: string }`
Institute	An educational instition where the students and alumni.	`export default interface IInstitute extends IMongoDocument {`

		name: string
		contact: {
		emails: string[]
		phone: string[]
		social: {
		facebook?: string
		instagram?: string
		x?: string
		quora?: string
		others?: [string, string][]
		}
		}
		admin: IMongoDocument["_id"]|IUser
		address: IMongoDocument["_id"]|IAddress
		}

Post	The content shared by the user globally.	interface IPost extends IMongoDocument {
		user: IMongoDocument["_id"]|IUser
		content: {
		text: string
		media?: string[]
		}
		createdAt: Time
		institute?:
		IMongoDocument["_id"]|IInstitute
		}

News	The news shared by the admin.	interface INews extends IMongoDocument {
		title: string
		description?: string
		createdAt: Time
		}

Interaction	The interaction like likes and comments on a post.	interface IInteraction extends IMongoDocument
		{
		post: IMongoDocument["_id"]|IPost
		user: IMongoDocument["_id"]|IUser
		type: String|InteractionType
		comment?: string
		createdAt: Time
		}

VALUED OBJECTS

VALUE OBJECTS	DESCRIPTION	PSEUDO CODE
Skill	The specific skill or expertise mentioned in a user's profile which remains	export default interface ISkill extends IMongoDocument { name: string

	constant once added.	`}`

Education	Information about a user's education including the institution, degree and graduation date which remains constant once added.	`export default interface IEducation extends IMongoDocument {`
		`  institute: IMongoDocument["_id"]|IInstitute`
		`  type: EducationTypes|string`
		`  joined: Time`
		`  completion: {`
		`  isCurrent: boolean`
		`  completed?: Time`
		`  }`
		`  remarks?: string`
		`}`

Achievements	Information about a user's achievement which remains constant once added.	`export default interface IAchievement extends IMongoDocument {`
		`  info: string`
		`  description?: string`
		`  documents: string[]`
		`  createdAt: Time`
		`}`

Work Experience	Information about a user's work experience including the company details, start date and end date which remains constant once added.	`export default interface IWork extends IMongoDocument {`
		`  company: IMongoDocument["_id"]|ICompany`
		`  from: Time`
		`  isCurrent: boolean`
		`  to?: Time`
		`  name: string`
		`}`

AGGREGATES

AGGREGATE	DESCRIPTION	ENTITIES	VALUED OBJECTS	PSEUDO CODE
User Profile	The profile of the user once the user signs in or logs in into the app.	User	Skill, Education, Achievements, Work Experience	Import getValue function from "utils/object" Import Router from "express" Create an instance of Router and assign it to app Create an instance of UserHandler and assign it to handler Define a POST route "/create" on app with middleware functions: verifyToken(), verifyBody(["role"]), and an async request handler function (req, res). Inside the request handler function: Retrieve keys, values, and session

				from res.locals Find a user by ID using the session information If no user is found: Send a 404 status response with an error message (MISSING_AUTH_HEADER) Create an address using the address data from the request body Create a new profile with the address ID If "education" is included in keys: Set the education field of the profile with the corresponding value If "achievements" is included in keys: Set the achievements field of the profile with the corresponding value If "skills" is included in keys: Set the skills field of the profile with the corresponding value If "workExperience" is included in keys: Set the workExperience field of the profile with the corresponding value Save the profile If the profile is not saved successfully: Send a 400 status response with an error message (STATUS_404) Set the profile of the user to the newly created profile Save the updated user If the user is not saved successfully: Send a 400 status response with an error message (STATUS_404) Send a 200 status response with the updated user data
Connection Network	The mutual connections between different users.	User, Connection	Connection Request	Define a POST route "/create" on the application with middleware function verifyToken() and an async request handler function (req, res). Inside the request handler function: Extract userIds from the request body Create a new connection with the provided userIds If the connection creation fails: Send a 404 status response with an error message (STATUS_404) Send a 200 status response with the created connection data
Job Application	The process in which the alumni posts	Job, User	Job Application	Import JobHandler from "@handlers/job"; Import logger from

	and users contact the alumni.			"@server/logger/winston"; Import verifyBody and verifyToken middleware functions from "@server/middleware/verify"; Import Job and Company models from "@server/models/job"; Import IUser type from "@types_/user"; Import getValue function from "@utils/object"; Import Router from "express"; Create a new instance of Router and assign it to app. Create a new instance of JobHandler and assign it to handler. Define a POST route "/create" on app with the following middleware functions: - verifyToken() - verifyBody(["title", "description", "company", "experienceYears"]) Define an async request handler function (req, res): Extract keys, values, and session from res.locals. Find a company by name using the provided company name. If the company doesn't exist: Create a new company with the provided name. If the company creation fails: Send a 404 status response with an error message (STATUS_404). Create a new job instance with the following properties: - from: ID of the user from the session - title: Value extracted from keys and values for "title" - description: Value extracted from keys and values for "description" - company: Retrieved or newly created company object - experienceYears: Value extracted from keys and values for "experienceYears" - Optionally, set skills, endsAt, and isCompleted properties based on their presence in keys and values. Set the skills property of the job with the skills from the request body.

				Save the job instance. If job saving fails: Send a 400 status response with an error message (STATUS_404). Send a 200 status response with the job data. Export app as the default module.
Search	The aggregate which is used for users and jobs	User, Jobs	Profile	Define a GET route "/search/:title" on the application with the following middleware functions: - verifyToken() - verifyParams(["title"]) Define an async request handler function (req, res): Extract keys and values from res.locals. Extract the title from keys and values using getValue function. If title doesn't exist: Send a 400 status response with an error message (STATUS_404). Find jobs using Job model where the title matches the provided title case-insensitively. Populate the "company" and "from" fields in the job documents. If no jobs are found: Send a 404 status response with an error message (STATUS_404). Send a 200 status response with the found jobs. Define a GET route "/get-all" on the application with the verifyToken() middleware function. Define an async request handler function (_, res): Extract session from res.locals. Find all users except the current user. If no users are found: Send a 404 status response with an error message (STATUS_404). Send a 200 status response with the found users.
Post Aggregate	The aggregation of the	Post, Interaction, User	None	Create a new instance of Router and assign it to app. Create a Multer instance and assign it

	interactions to posts and posts.			to multer. Create a new instance of PostHandler and assign it to handler. Define a POST route "/create" on app with the following middleware functions: - verifyToken() - multer.array("content.media") - verifyBody(["content.text"]) Define an async request handler function (req, res): Extract keys, values, and session from res.locals. Extract user ID from the session. Extract files from the request. Initialize an empty array fileUrls. Iterate over each file: - Generate a unique file name based on user ID and original file name. - Download the file and get its URL. - If URL exists, push it to fileUrls array. Create a new post with user ID and content. If post creation fails, send a 404 status response with an error message (STATUS_404). Send a 200 status response with the created post. Define a PUT route "/:post/interact/:type" on app with the verifyToken() and verifyParams(["post", "type"]) middleware functions. Define an async request handler function (req, res): Extract keys, values, and session from res.locals. Check if the provided interaction type is valid. Extract request body. Check if comment is required for a comment interaction type and validate it. Create a new interaction based on the request parameters. If interaction creation fails, send a 404 status response with an error message (STATUS_404). Send a 200 status response with the created interaction or deleted

				interaction (if already liked). Define a GET route "/:post/interactions/:type" on app with the verifyToken() and verifyParams(["post"]) middleware functions. Define an async request handler function (req, res): Extract keys and values from res.locals. Find interactions based on post ID and optionally interaction type. Populate user field in interactions. Send a 200 status response with the interactions. Define a DELETE route "/:id" on app with the verifyToken() and verifyParams(["id"]) middleware functions. Define an async request handler function (req, res): Extract keys and values from res.locals. Delete all interactions associated with the post. Delete the post itself. If post deletion fails, send a 404 status response with an error message (STATUS_404). Send a 200 status response with the deleted post. Export app as the default module.

SERVICES

SERVICE	DESCRIPTION	AGGREGATE
User Profile Service	Manages operations related to user profiles, aggregating entities such as User, Skill, EducationInfo, and value objects like Address, EmailAddress, and PhoneNumber.	User, User Profile
Connection Service	Handles actions related to connections, aggregating entities like User and Connection within the context of a user's network.	Connection, Connection Request
Job Service	Facilitates job-related operations, aggregating entities like User, JobPosting and Application within the context of job postings.	Job, User
Posting Service	Facilitates post-related operations, aggregating entities like User, Post and Interaction within the	User, Post, Interaction

	context of posts.	
Seaching Service	Provides functionality for searching and discovering users, jobs and other relevant content, aggregating various entities within search results.	Search, User, Job

REPOSITORIES

- MongoDB
- Local Storage