System Design Document for the African Impact Challenge Project

CSCC01 Summer 2021
Team AgileCats

Kenneth Daniel
Jenny Ho
Alexander Efimov
Xin Ya Xu
Richard Zheng
HongKang Yu

Table of Contents

CRC Cards

Implemented in sprint 1:

DTOs

User Signup DTO	
Parent: None Subclass: None	
Responsibilities: - Get user info for signup - Validate input data	Collaborators: User controller Auth controller

User Login DTO	
Parent: None Subclass: None	
Responsibilities: - Get user info for login - Validate input data	Collaborators User controller Auth controller User service Auth service

Mentor profile DTO	
Parent: None Subclass: None	
Responsibilities: - Validate input data	Collaborators: Mentor controller

Partner profile DTO	
Parent: None Subclass: None	

Responsibilities: - Validate input data	Collaborators: Partner profile controller

Participant profile DTO	
Parent: None Subclass: None	
Responsibilities: - Validate input data	Collaborators: Participant profile controller

Services

User Service	
Parent: None Subclass: None	
Responsibilities: Perform database operations related to users: - Find user by ID - Return all users - Create user - Update user - Delete user User model	Collaborators: User signup DTO User model User (interface)

Mentor Profile Service	
Parent: None Subclass: None	
Responsibilities: Perform database operations related to mentors: - Find by ID - Return all mentors - Create - Update - Delete Mentor profile model User model	Collaborators: Mentor profile signup DTO Mentor profile model User model User (interface) Mentor profile (interface)

Participant Profile Service	
Parent: None Subclass: None	
Responsibilities: Perform database operations related to participants: - Find by ID - Return all participants - Create - Update - Delete Participant profile model User model	Collaborators: Participant Signup DTO Participant model User model User (interface) Participant (interface)

Partner Profile Service	
Parent: None Subclass: None	
Responsibilities: Perform database operations related to partner: - Find by ID - Return all partners - Create - Update - Delete Partner profile model User model	Collaborators: Partner Signup DTO Partner model User model User (interface) Partner (interface)

Authentication Service	
Parent: None Subclass: None	
Responsibilities:Handle requests related to authentication: - Sign up - Log in - Creating session tokens/cookies	Collaborators: User model User signup DTO

User model	

Models

User Model	
Parent: None Subclass: None	
Responsibilities: - Translate between schemas and database documents - Provide API for interacting with documents	Collaborators: User (interface)

Partner profile Model	
Parent: None Subclass: None	
Responsibilities: - Translate between schemas and database documents - Provide API for interacting with documents	Collaborators: Partner profile (interface)

Participant profile Model	
Parent: None Subclass: None	
Responsibilities: - Translate between schemas and database documents - Provide API for interacting with documents	Collaborators: Participant profile (interface)

Mentor profile Model	
Parent: None Subclass: None	
Responsibilities:	Collaborators:

- Translate between schemas and database documents - Provide API for interacting with documents	Mentor profile (interface)

Controllers

User Controller	
Parent: None Subclass: None	
Responsibilities: User service	Collaborators: User (interface) User signup DTO User service

Index Controller	
Parent: None Subclass: None	
Responsibilities:	Collaborators:

Auth Controller	
Parent: None Subclass: None	
Responsibilities: Auth service	Collaborators: User (interface) User signup DTO Auth service

Mentor profile controller	
Parent: None Subclass: None	
Responsibilities: Mentor profile service	Collaborators: Mentor profile (interface) Mentor signup DTO

Partner profile controller	

Parent: None Subclass: None	
Responsibilities: Partner profile service	Collaborators: Partner profile (interface) Partner signup DTO

Participant profile controller	
Parent: None Subclass: None	
Responsibilities: Participant profile service	Collaborators: Participant profile (interface) Participant signup DTO

Implemented in sprint 2:

DTOs

Update Profile DTO	
Parent: None Subclass: None	
Responsibilities: - Get user info for updating profile - Validate input data - Has email, password, firstName, lastName, aboutMe, proBackground, skills, companyName, companyDescription	Collaborators: - User Controller

Media DTO	
Parent: None Subclass: None	
Responsibilities: - Validate media info input data - Has uploadUser, viewAccess, type, description, title, tags	Collaborators:

Base Post DTO	
Parent: None Subclass: None	
Responsibilities: - Validate post info input data - Has postUser, viewAccess, content, description, title, tags, contentType	Collaborators:

Services

Authentication Service	
Parent: None Subclass: None	
Responsibilities: Perform database operations related to users: - Send password reset link to users email - Change a users password - Check if reset password token is valid User model Token model	Collaborators: User model Token model Auth (interface)

Models

Token Model	
Parent: None Subclass: None	
Responsibilities: - Translate between schemas and database documents - Provide API for interacting with documents - Has userId, token and createdAt date	Collaborators:

Media Model	
Parent: None Subclass: None	
Responsibilities: - Translate between schemas and database documents - Provide API for interacting with documents - Has uploadUser, uploadDateTime, viewAccess, file, description, title, tags, type	Collaborators:

Post Model	
Parent: None Subclass: None	
Responsibilities: - Translate between schemas and database documents - Provide API for interacting with documents - Has postUser, postDateTime, viewAccess, tags, description, title, tags, contentType	Collaborators:

Controllers

Authentication Controller	
Parent: None Subclass: None	
Responsibilities: Token service	Collaborators: User (interface) User signup DTO User service

MediaController	
Parent: None Subclass: None	
Responsibilities: Perform database operations related to media: - Find media by ID - Return all medias - Create media Media model	Collaborators: Model (interface) Model DTO

PostController	
Parent: None Subclass: None	
Responsibilities: Perform database operations related to media: - Find post by ID - Return all posts - Create post Media model	Collaborators: Post(interface) Post DTO

Implemented in sprint 3:

DTOs

Company DTO	
Parent: None Subclass: None	
Responsibilities: - Validate input data for company info - Has companyName, founderName, website, industry, companyType, aboutUs, specialties, users	Collaborators: - Company Controller

Deliverable DTO	
Parent: None Subclass: None	
Responsibilities: - Validate deliverable info input data - Has title, description, dueDate	Collaborators: - Deliverable Controller

Comment DTO	
Parent: None Subclass: None	
Responsibilities: - Validate comment info input data - Has userId, content	Collaborators: - Comment Controller

Services

Company Service	
Parent: None Subclass: None	
Responsibilities: Perform database operations related to companies: - Find all companies - Find companies by Id - Create a company - Update a company - Add a user to a company - Remove a user from a company - Delete a company Company model	Collaborators: Company model Company (interface)

Models

Company Model	
Parent: None Subclass: None	
Responsibilities: - Translate between schemas and database documents - Provide API for interacting with documents - Has companyName, founderName, website, industry, companyType, aboutUs, specialties, users	Collaborators:

Deliverable Model	
Parent: None Subclass: None	
Responsibilities: - Translate between schemas and database documents - Provide API for interacting with documents	Collaborators:

- Has createdDate, dueDate, title, description	

Comment Model	
Parent: None Subclass: None	
Responsibilities: - Translate between schemas and database documents - Provide API for interacting with documents - Has userId, postId, createdAt, content	Collaborators:

Controllers

Company Controller	
Parent: None Subclass: None	
Responsibilities: Company service	Collaborators: Company (interface) Company DTO Company service

Deliverable Controller	
Parent: None Subclass: None	
Responsibilities: Perform database operations related to deliverable: - Find deliverable by ID - Return all deliverables - Create deliverable Deliverable model	Collaborators: Deliverable (interface) Deliverable DTO

Comment Controller	
Parent: None Subclass: None	
Responsibilities: Perform database operations related to comment: - Find comment by ID - Return all comments of a post - Create comment for a post - Delete comment of a post Comment model	Collaborators: Comment (interface) Comment DTO

Design for future plans:

Message	
Parent: None Subclass: None	
Responsibilities: - Sender (user) - Receiver (user) - Timestamp (at send time) - Message contents	Collaborators: User

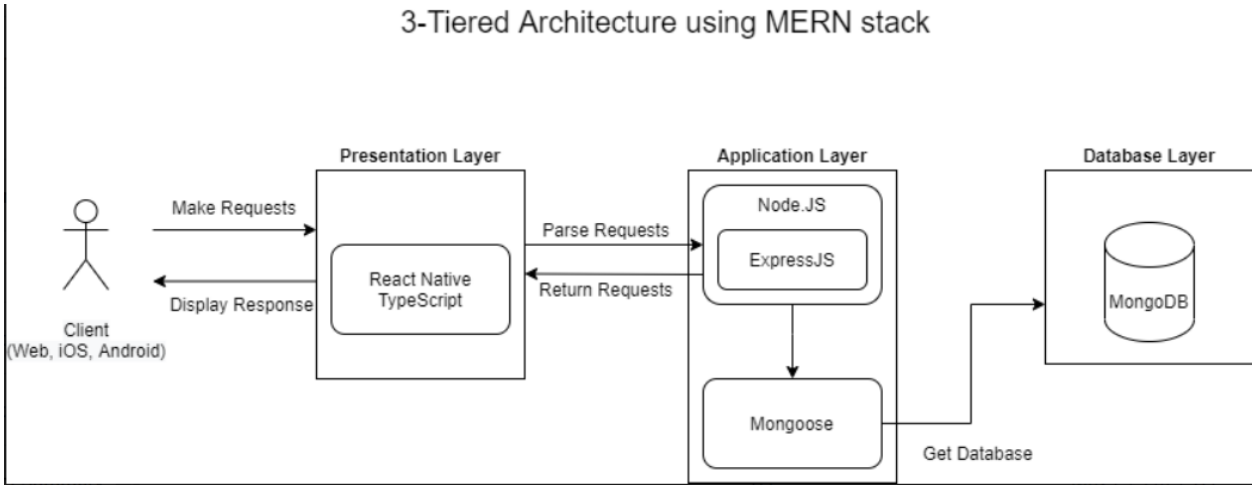
DeliverableSubmission	
Parent: None Subclass: None	
Responsibilities: - Knows which deliverable - User who submitted - Deliverable contents - Submission date	Collaborators: User Deliverables

Software Architecture Diagram (Three-Tiered Architecture)

Three-Tiered Architecture: https://www.linuxjournal.com/article/3508

System Interaction with the Environment

Our system is designed to be able to operate across all operating systems. Our project utilizes the MongoDB database with NodeJS, ExpressJS, and React Native frameworks. Users of our system should install the required software in order to run the project locally. The instructions can be found in the README file in our github repo (https://github.com/UTSCCSCC01/project-agilecats).



System Decomposition

Client Layer (React Native):

- [Login Component] Invalid user input (incorrect credentials) during login will be handled with an on-screen alert
- [Login Auth Component] Invalid API requests to nodeJS will be logged as errors in the console and the response will not be displayed.
- [Signup Component] Invalid user input (incorrect format) during signup will be handled with an on-screen alert

Application Layer (Node.js, Express.js):

- Handle API requests
    - Validate input
    - Perform database operation if validation successful
    - Return result

Database Layer (MongoDB):

- User Model

- Profile models:
    - Mentor profile
    - Participant profile
    - Investor profile

API requests return code 400 with a helpful status message when the input is invalid. For operations that deal with a specific object by id, if the object cannot be found, the API returns 409.
The front end currently expects 200, and anything else results in an error. In the future, the front end should be made to handle different kinds of errors differently to improve the user experience. e.g. currently, all non-200 codes on the login page result in a potentially-confusing ‘incorrect password’ message.