

# 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

<b>CRC Cards</b>	<b>3</b>
Implemented in sprint 1:	3
DTOs	3
Services	4
Models	6
Controllers	7
Implemented in sprint 2:	10
DTOs	11
Services	11
Models	12
Controllers	12
Implemented in sprint 3:	14
DTOs	14
Services	15
Models	15
Controllers	16
Implemented in sprint 4:	17
DTOs	17
Models	18
Controllers	19
 <b>Software Architecture Diagram (Three-Tiered Architecture)</b>	 <b>20</b>
System Interaction with the Environment	20
System Decomposition	20

## CRC Cards

Implemented in sprint 1:

### DTOs

User Signup DTO	
Parent: None Subclass: None	
Responsibilities: <ul style="list-style-type: none"> <li>- Get user info for signup</li> <li>- Validate input data</li> </ul>	Collaborators: User controller Auth controller

User Login DTO	
Parent: None Subclass: None	
Responsibilities: <ul style="list-style-type: none"> <li>- Get user info for login</li> <li>- Validate input data</li> </ul>	Collaborators User controller Auth controller User service Auth service

Mentor profile DTO	
Parent: None Subclass: None	
Responsibilities: <ul style="list-style-type: none"> <li>- Validate input data</li> </ul>	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	Collaborators: Mentor profile signup DTO Mentor profile model User model User (interface) Mentor profile (interface)

- Delete Mentor profile model User model	
--	--

Participant Profile Service	
Parent: None Subclass: None	
Responsibilities: Perform database operations related to participants: <ul style="list-style-type: none"> <li>- Find by ID</li> <li>- Return all participants</li> <li>- Create</li> <li>- Update</li> <li>- Delete</li> </ul> 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: <ul style="list-style-type: none"> <li>- Find by ID</li> <li>- Return all partners</li> <li>- Create</li> <li>- Update</li> <li>- Delete</li> </ul> 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:	Collaborators: User model

<ul style="list-style-type: none"> <li>- Sign up</li> <li>- Log in</li> <li>- Creating session tokens/cookies</li> </ul> User model	User signup DTO
---	-----------------

## Models

User Model	
Parent: None Subclass: None	
Responsibilities: <ul style="list-style-type: none"> <li>- Translate between schemas and database documents</li> <li>- Provide API for interacting with documents</li> </ul>	Collaborators: User (interface)

Partner profile Model	
Parent: None Subclass: None	
Responsibilities: <ul style="list-style-type: none"> <li>- Translate between schemas and database documents</li> <li>- Provide API for interacting with documents</li> </ul>	Collaborators: Partner profile (interface)

Participant profile Model	
Parent: None Subclass: None	
Responsibilities: <ul style="list-style-type: none"> <li>- Translate between schemas and database documents</li> <li>- Provide API for interacting with documents</li> </ul>	Collaborators: Participant profile (interface)

Mentor profile Model	
Parent: None	

Subclass: None	
Responsibilities: <ul style="list-style-type: none"> <li>- Translate between schemas and database documents</li> <li>- Provide API for interacting with documents</li> </ul>	Collaborators: 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: <ul style="list-style-type: none"> <li>- Get user info for updating profile</li> <li>- Validate input data</li> <li>- Has email, password, firstName, lastName, aboutMe, proBackground, skills, companyName, companyDescription</li> </ul>	Collaborators: <ul style="list-style-type: none"> <li>- User Controller</li> </ul>

Media DTO	
Parent: None Subclass: None	
Responsibilities: <ul style="list-style-type: none"> <li>- Validate media info input data</li> <li>- Has uploadUser, viewAccess, type, description, title, tags</li> </ul>	Collaborators:

Base Post DTO	
Parent: None Subclass: None	
Responsibilities: <ul style="list-style-type: none"> <li>- Validate post info input data</li> <li>- Has postUser, viewAccess, content, description, title, tags, contentType</li> </ul>	Collaborators:

## Services

Authentication Service	
Parent: None Subclass: None	
Responsibilities: Perform database operations related to users: <ul style="list-style-type: none"> <li>- Send password reset link to users email</li> <li>- Change a users password</li> <li>- Check if reset password token is valid</li> </ul> User model Token model	Collaborators: User model Token model Auth (interface)

## Models

Token Model	
Parent: None Subclass: None	
Responsibilities: <ul style="list-style-type: none"> <li>- Translate between schemas and database documents</li> <li>- Provide API for interacting with documents</li> <li>- Has userId, token and createdAt date</li> </ul>	Collaborators:

Media Model	
Parent: None Subclass: None	
Responsibilities: <ul style="list-style-type: none"> <li>- Translate between schemas and database documents</li> <li>- Provide API for interacting with documents</li> <li>- Has uploadUser, uploadDateTime, viewAccess, file, description, title, tags, type</li> </ul>	Collaborators:

Post Model	
Parent: None Subclass: None	
Responsibilities: <ul style="list-style-type: none"> <li>- Translate between schemas and database documents</li> <li>- Provide API for interacting with documents</li> <li>- Has postUser, postDateTime, viewAccess, tags, description, title, tags, contentType</li> </ul>	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: <ul style="list-style-type: none"> <li>- Find media by ID</li> <li>- Return all medias</li> <li>- Create media</li> </ul> Media model	Collaborators: Model (interface) Model DTO

PostController	
Parent: None Subclass: None	
Responsibilities: Perform database operations related to media: <ul style="list-style-type: none"><li>- Find post by ID</li><li>- Return all posts</li><li>- Create post</li></ul> Media model	Collaborators: Post(interface) Post DTO

Implemented in sprint 3:

## DTOs

Company DTO	
Parent: None Subclass: None	
Responsibilities: <ul style="list-style-type: none"> <li>- Validate input data for company info</li> <li>- Has companyName, founderName, website, industry, companyType, aboutUs, specialties, users</li> </ul>	Collaborators: <ul style="list-style-type: none"> <li>- Company Controller</li> </ul>

Deliverable DTO	
Parent: None Subclass: None	
Responsibilities: <ul style="list-style-type: none"> <li>- Validate deliverable info input data</li> <li>- Has title, description, dueDate</li> </ul>	Collaborators: <ul style="list-style-type: none"> <li>- Deliverable Controller</li> </ul>

Comment DTO	
Parent: None Subclass: None	
Responsibilities: <ul style="list-style-type: none"> <li>- Validate comment info input data</li> <li>- Has userId, content</li> </ul>	Collaborators: <ul style="list-style-type: none"> <li>- Comment Controller</li> </ul>

## Services

Company Service	
Parent: None Subclass: None	
Responsibilities: Perform database operations related to companies: <ul style="list-style-type: none"> <li>- Find all companies</li> <li>- Find companies by Id</li> <li>- Create a company</li> <li>- Update a company</li> <li>- Add a user to a company</li> <li>- Remove a user from a company</li> <li>- Delete a company</li> </ul> Company model	Collaborators: Company model Company (interface)

## Models

Company Model	
Parent: None Subclass: None	
Responsibilities: <ul style="list-style-type: none"> <li>- Translate between schemas and database documents</li> <li>- Provide API for interacting with documents</li> <li>- Has companyName, founderName, website, industry, companyType, aboutUs, specialties, users</li> </ul>	Collaborators:

Deliverable Model	
Parent: None Subclass: None	
Responsibilities: <ul style="list-style-type: none"> <li>- Translate between schemas and database documents</li> <li>- Provide API for interacting with documents</li> </ul>	Collaborators:

- Has createdDate, dueDate, title, description	
--	--

Comment Model	
Parent: None Subclass: None	
Responsibilities: <ul style="list-style-type: none"> <li>- Translate between schemas and database documents</li> <li>- Provide API for interacting with documents</li> <li>- Has userId, postId, createdAt, content</li> </ul>	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: <ul style="list-style-type: none"> <li>- Find deliverable by ID</li> <li>- Return all deliverables</li> <li>- Create deliverable</li> </ul> Deliverable model	Collaborators: Deliverable (interface) Deliverable DTO

Comment Controller	
Parent: None Subclass: None	
Responsibilities: Perform database operations related to comment: <ul style="list-style-type: none"><li>- Find comment by ID</li><li>- Return all comments of a post</li><li>- Create comment for a post</li><li>- Delete comment of a post</li></ul> Comment model	Collaborators: Comment (interface) Comment DTO



## Implemented in sprint 4:

## DTOs

GradeDeliverableDTO	
Parent: None Subclass: None	
Responsibilities: <ul style="list-style-type: none"> <li>- Validate input data for grading info</li> <li>- Has grade, feedback and graderId</li> </ul>	Collaborators: <ul style="list-style-type: none"> <li>- Submission Controller</li> </ul>

PostMessageDTO	
Parent: None Subclass: None	
Responsibilities: <ul style="list-style-type: none"> <li>- Validate input data for post message info</li> <li>- Has userId and message</li> </ul>	Collaborators: <ul style="list-style-type: none"> <li>- Message Controller</li> </ul>

CreateConversationDTO	
Parent: None Subclass: None	
Responsibilities: <ul style="list-style-type: none"> <li>- Validate conversation info input data</li> <li>- Has userId and receiverId</li> </ul>	Collaborators: <ul style="list-style-type: none"> <li>- Message Controller</li> </ul>

AddUserToConversationDTO	
Parent: None Subclass: None	
Responsibilities: <ul style="list-style-type: none"> <li>- Validate add user to conversation info input data</li> <li>- Has userId to add</li> </ul>	Collaborators: <ul style="list-style-type: none"> <li>- Message Controller</li> </ul>

## Models

Submission Model	
Parent: None Subclass: None	
Responsibilities: <ul style="list-style-type: none"> <li>- Translate between schemas and database documents</li> <li>- Provide API for interacting with documents</li> <li>- Has companyId, deliverableId, fileId, fileName, grade, feedback, onTime, isGraded, graderId and timestamp</li> </ul>	Collaborators:

Conversation Model	
Parent: None Subclass: None	
Responsibilities: <ul style="list-style-type: none"> <li>- Translate between schemas and database documents</li> <li>- Provide API for interacting with documents</li> <li>- Has members (list of members of conversation that contains userId, firstName and lastName), lastMessage, numOfMembers and timestamps</li> </ul>	Collaborators:

Message Model	
Parent: None Subclass: None	
Responsibilities: <ul style="list-style-type: none"> <li>- Translate between schemas and database documents</li> <li>- Provide API for interacting with documents</li> <li>- Has conversationId, userId, firstName, lastName, message and date</li> </ul>	Collaborators:

## Controllers

Submission Controller	
Parent: None Subclass: None	
Responsibilities: Perform database operations related to submissions: <ul style="list-style-type: none"> <li>- Submit submission for a deliverable</li> <li>- Grade a submission</li> <li>- Find submission by Id</li> <li>- Return all submissions</li> <li>- Return submission file to download</li> <li>- Return all submission by deliverable</li> <li>- Get grade and feedback of a submission</li> </ul> Submission model	Collaborators: Submission (interface) GradeDeliverableDTO

Message Controller	
Parent: None Subclass: None	
Responsibilities: Perform database operations related to message and conversation: <ul style="list-style-type: none"> <li>- Create a conversation</li> <li>- Get all conversation of user</li> <li>- Get all messages from conversation</li> <li>- Create a message</li> <li>- Add a user to an existing conversation</li> <li>- Delete a conversation</li> </ul> Message model Conversation model	Collaborators: Message (interface) Conversation (interface) PostMessageDTO CreateConversationDTO AddUserToConversationDTO

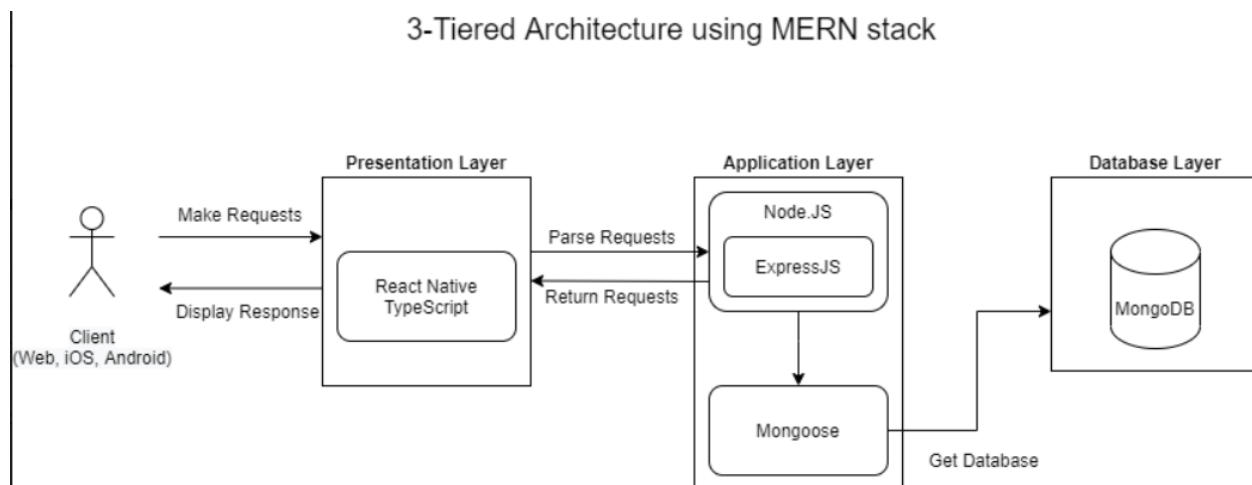
# Software Architecture Diagram (Three-Tiered Architecture)

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.