Project:

Team Members: Abdulrahman Alaraj, Chirag Kamat, Ganesh ByrandurgaGopinath.
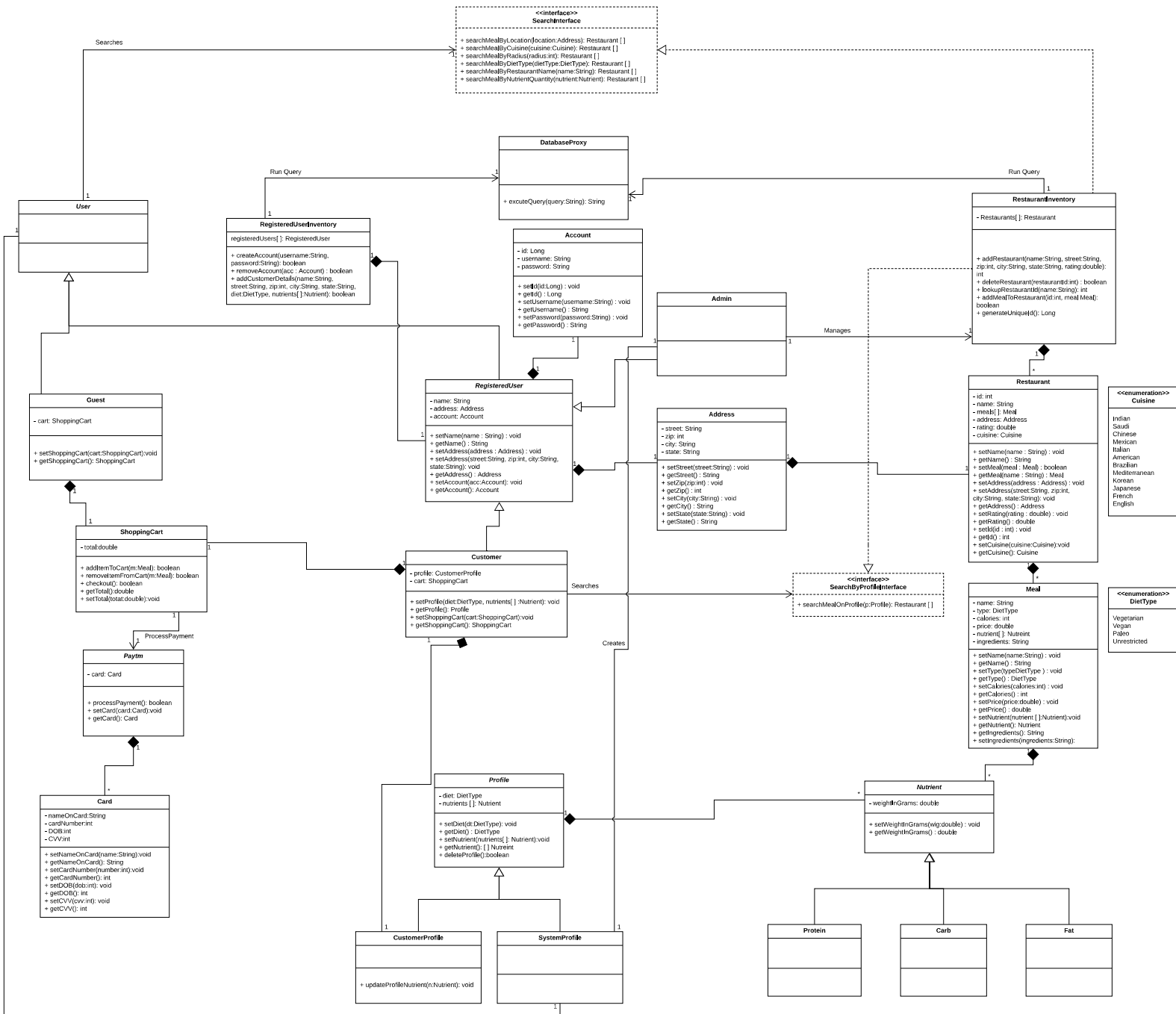
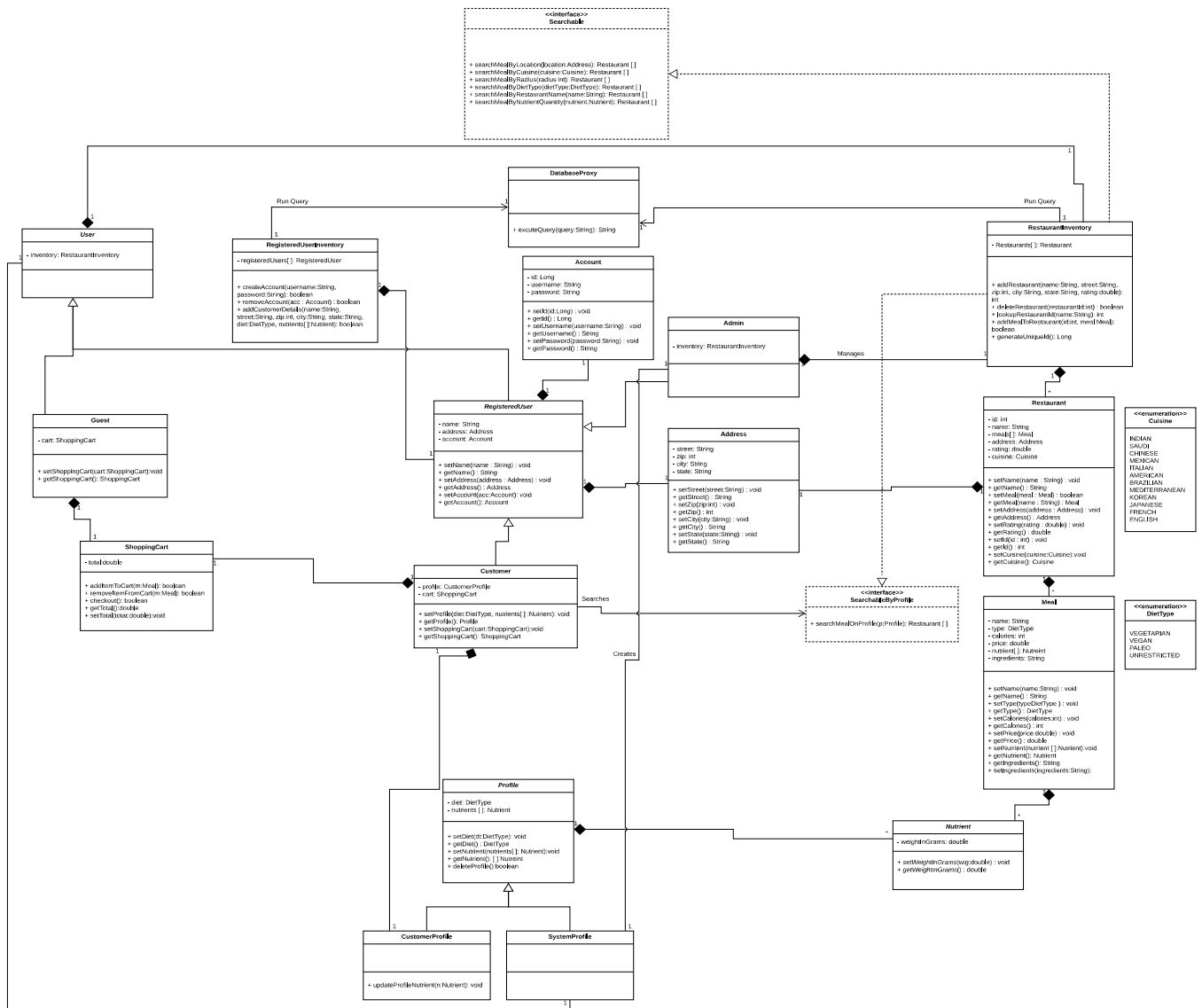**Team Number:** 40

Title: FitFoodie

Description: A web based system which recommends restaurants and allows the user to order food based on nutrition preferences\requirements.

Vision: Building a web based system by leveraging object oriented frameworks and applying object oriented design patterns.

Previous Class Diagram:

<<interface>> SearchInterface
+ searchMealByLocation(location:Address): Restaurant [ ]
+ searchMealByCuisine(cuisine:Cuisine): Restaurant [ ]
+ searchMealByRadius(radius:int): Restaurant [ ]
+ searchMealByDietType(dietType:DietType): Restaurant [ ]
+ searchMealByRestaurantName(name:String): Restaurant [ ]
+ searchMealByNutrientQuantity(nutrient:Nutrient): Restaurant [ ]

Searches

DatabaseProxy

+ excuteQuery(query:String): String

Run Query

Run Query

User

RegisteredUserInventory
registeredUsers[ ]: RegisteredUser
+ createAccount(username:String, password:String): boolean
+ removeAccount(acc : Account) : boolean
+ addCustomerDetails(name:String, street:String, zip:int, city:String, state:String, diet:DietType, nutrients[ ]:Nutrient): boolean

Account
- id: Long
- username: String
- password: String
+ setId(id:Long) : void
+ getId() : Long
+ setUsername(username:String) : void
+ getUsername() : String
+ setPassword(password:String) : void
+ getPassword() : String

Admin

RestaurantInventory
- Restaurants[ ]: Restaurant
+ addRestaurant(name:String, street:String, zip:int, city:String, state: String, rating:double): int
+ deleteRestaurant(restaurantid:int) : boolean
+ lookupRestaurantId(name:String): int
+ addMealToRestaurant(id:int, meal:Meal): boolean
+ generateUniqueId(): Long

Manages

Guest
- cart: ShoppingCart
+ setShoppingCart(cart:ShoppingCart):void
+ getShoppingCart(): ShoppingCart

RegisteredUser
- name: String
- address: Address
- account: Account
+ setName(name : String) : void
+ getName() : String
+ setAddress(address : Address) : void
+ setAddress(street:String, zip:int, city:String, state:String): void
+ getAddress() :Address
+ setAccount(acc:Account): void
+ getAccount(): Account

Address
- street: String
- zip: int
- city: String
- state: String
+ setStreet(street:String) : void
+ getStreet() : String
+ setZip(zip:int) : void
+ getZip() : int
+ setCity(city:String) : void
+ getCity() : String
+ setState(state:String) : void
+ getState() : String

Restaurant
- id: int
- name: String
- meals[ ]: Meal
- address: Address
- rating: double
- cuisine: Cuisine
+ setName(name : String) : void
+ getName() : String
+ setMeal(meal : Meal) : boolean
+ getMeal(name : String) : Meal
+ setAddress(address : Address) : void
+ setAddress(street:String, zip:int, city:String, state:String): void
+ getAddress() : Address
+ setRating(rating : double) : void
+ getRating() : double
+ setId(id : int) : void
+ getId() : int
+ setCuisine(cuisine:Cuisine):void
+ getCuisine(): Cuisine

<<enumeration>> Cuisine
Indian
Saudi
Chinese
Mexican
Italian
American
Brazilian
Mediterranean
Korean
Japanese
French
English

ShoppingCart
- total:double
+ addItemToCart(m:Meal): boolean
+ removeItemFromCart(m:Meal): boolean
+ checkout(): boolean
+ getTotal():double
+ setTotal(totat:double):void

Customer
- profile: CustomerProfile
- cart: ShoppingCart
+ setProfile(diet:DietType, nutrients[ ] :Nutrient): void
+ getProfile(): Profile
+ setShoppingCart(cart:ShoppingCart):void
+ getShoppingCart(): ShoppingCart

Searches

<<interface>> SearchByProfileInterface
+ searchMealOnProfile(p:Profile): Restaurant [ ]

Creates

Meal
- name: String
- type: DietType
- calories: int
- price: double
- nutrient[ ]: Nutrient
- ingredients: String
+ setName(name:String) : void
+ getName() : String
+ setType(typeDietType) : void
+ getType() : DietType
+ setCalories(calories:int) : void
+ getCalories() : int
+ setPrice(price:double) : void
+ getPrice() : double
+ setNutrient(nutrient [ ]:Nutrient):void
+ getNutrient(): Nutrient
+ getIngredients(): String
+ setIngredients(ingredients:String):

<<enumeration>> DietType
Vegetarian
Vegan
Paleo
Unrestricted

ProcessPayment

Paytm
- card: Card
+ processPayment(): boolean
+ setCard(card:Card):void
+ getCard(): Card

Profile
- diet: DietType
- nutrients [ ]: Nutrient
+ setDiet(dt:DietType): void
+ getDiet() : DietType
+ setNutrient(nutrients[ ]: Nutrient):void
+ getNutrient(): [ ] Nutreint
+ deleteProfile():boolean

Nutrient
- weightInGrams: double
+ setWeightInGrams(wig:double) : void
+ getWeightInGrams() : double

Card
- nameOnCard:String
- cardNumber:int
- DOB:int
- CVV:int
+ setNameOnCard(name:String):void
+ getNameOnCard(): String
+ setCardNumber(number:int):void
+ getCardNumber(): int
+ setDOB(dob:int): void
+ getDOB(): int
+ setCVV(cvv:int): void
+ getCVV(): int

CustomerProfile
+ updateProfileNutrient(n:Nutrient): void

SystemProfile

Protein

Carb

Fat

Current Class Diagram:

<<interface>> Searchable

+ searchMealByLocation(location:Address): Restaurant [ ]
+ searchMealByCuisine(cuisine:Cuisine): Restaurant [ ]
+ searchMealByRadius(radius:int): Restaurant [ ]
+ searchMealByDietType(dietType:DietType): Restaurant [ ]
+ searchMealByRestaurantName(name:String): Restaurant [ ]
+ searchMealByNutrientQuantity(nutrient:Nutrient): Restaurant [ ]

DatabaseProxy

+ excuteQuery(query:String): String

Run Query

Run Query

User

- inventory: RestaurantInventory

RegisteredUserInventory

- registeredUsers[ ]: RegisteredUser

+ createAccount(username:String, password:String): boolean
+ removeAccount(acc : Account) : boolean
+ addCustomerDetails(name:String, street:String, zip:int, city:String, state:String, diet:DietType, nutrients[ ]:Nutrient): boolean

Account

- id: Long
- username: String
- password: String

+ setId(id:Long) : void
+ getId() : Long
+ setUsername(username:String) : void
+ getUsername() : String
+ setPassword(password:String) : void
+ getPassword() : String

Admin

- inventory: RestaurantInventory

Manages

RestaurantInventory

- Restaurants[ ]: Restaurant

+ addRestaurant(name:String, street:String, zip:int, city:String, state:String, rating:double): int
+ deleteRestaurant(restaurantId:int) : boolean
+ lookupRestaurantId(name:String): int
+ addMealToRestaurant(id:int, meal:Meal): boolean
+ generateUniqueId(): Long

Guest

- cart: ShoppingCart

+ setShoppingCart(cart:ShoppingCart):void
+ getShoppingCart(): ShoppingCart

RegisteredUser

- name: String
- address: Address
- account: Account

+ setName(name : String) : void
+ getName() : String
+ setAddress(address : Address) : void
+ getAddress() : Address
+ setAccount(acc:Account): void
+ getAccount(): Account

Address

- street: String
- zip: int
- city: String
- state: String

+ setStreet(street:String) : void
+ getStreet() : String
+ setZip(zip:int) : void
+ getZip() : int
+ setCity(city:String) : void
+ getCity() : String
+ setState(state:String) : void
+ getState() : String

Restaurant

- id: int
- name: String
- meal[ ]: Meal
- address: Address
- rating: double
- cuisine: Cuisine

+ setName(name : String) : void
+ getName() : String
+ setMeal(meal : Meal) : boolean
+ getMeal(name : String) : Meal
+ setAddress(address : Address) : void
+ getAddress() : Address
+ setRating(rating : double) : void
+ getRating() : double
+ setId(id : int) : void
+ getId() : int
+ setCuisine(cuisine:Cuisine):void
+ getCuisine() : Cuisine

<<enumeration>> Cuisine

INDIAN
SAUDI
CHINESE
MEXICAN
ITALIAN
AMERICAN
BRAZILIAN
MEDITERRANEAN
KOREAN
JAPANESE
FRENCH
ENGLISH

ShoppingCart

- total:double

+ addItemToCart(m:Meal): boolean
+ removeItemFromCart(m:Meal): boolean
+ checkout(): boolean
+ getTotal():double
+ setTotal(total:double):void

Customer

- profile: CustomerProfile
- cart: ShoppingCart

+ setProfile(diet:DietType, nutrients[ ]:Nutrient): void
+ getProfile(): Profile
+ setShoppingCart(cart:ShoppingCart):void
+ getShoppingCart(): ShoppingCart

Searches

Creates

<<interface>> SearchableByProfile

+ searchMealOnProfile(p:Profile): Restaurant [ ]

Meal

- name: String
- type: DietType
- calories: int
- price: double
- nutrient[ ]: Nutrient
- ingredients: String

+ setName(name:String) : void
+ getName() : String
+ setType(type:DietType) : void
+ getType() : DietType
+ setCalories(calories:int) : void
+ getCalories() : int
+ setPrice(price:double) : void
+ getPrice() : double
+ setNutrient(nutrient [ ]:Nutrient):void
+ getNutrient(): Nutrient
+ getIngredients(): String
+ setIngredients(ingredients:String):

<<enumeration>> DietType

VEGETARIAN
VEGAN
PALEO
UNRESTRICTED

Profile

- diet: DietType
- nutrients [ ]: Nutrient

+ setDiet(dt:DietType): void
+ getDiet() : DietType
+ setNutrient(nutrients[ ]: Nutrient):void
+ getNutrient(): [ ] Nutrient
+ deleteProfile():boolean

Nutrient

- weightInGrams: double

+ setWeightInGrams(wig:double) : void
+ getWeightInGrams() : double

CustomerProfile

+ updateProfileNutrient(n:Nutrient): void

SystemProfile

Summary:

After developing a good understanding of the underpinnings of the system which the team aimed at developing, and after successfully completing the planning, analysis and design phases, the team has started the implementation phase. This phase is comprised for writing the code for the back-end using Java and writing the code for Django framework.

The team is making some good progress in the implementation phase, as nearly all classes are fully implemented. The team has also made some changes to the class diagram in order to comply with the given feedback for project part 2.

Work Breakdown:

Abdulrahman Alaraj: Implemented the class diagram and made necessary modifications to comply with project part 2 feedback.

Chirag Kamat: Implemented the class diagram and made necessary modifications to comply with project part 2 feedback.

Ganesh ByrandurgaGopinath: Worked on starting with Django application and was able to run few webpages locally.

GitHub Graphs:

0x90-n: Abdulrahman Alaraj
Ganeshbg: Ganesh ByrandurgaGopinath
ckamat1:Chirag Kamat
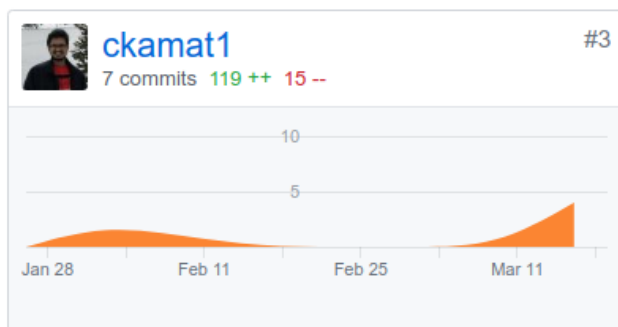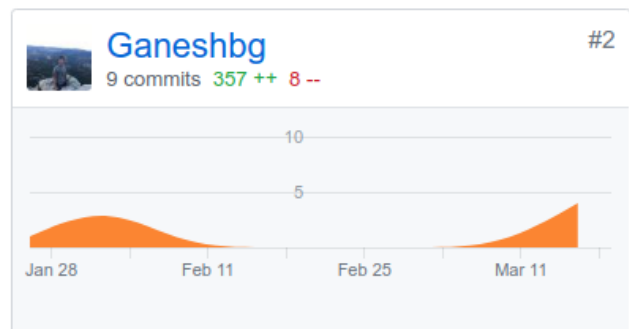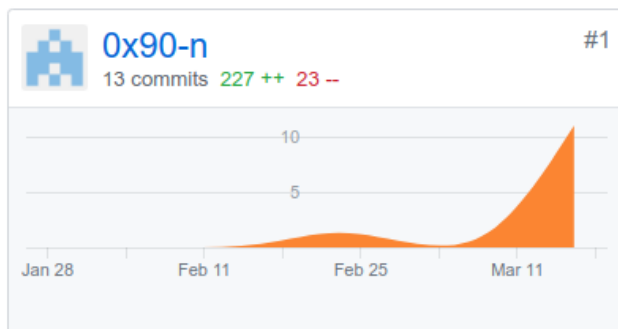
Estimate Remaining Effort:

Developing front-end web pages : 15 hours
Developing complete back-end system synchronous to the front-end : 40 hours
Integrating front end and back-end, testing the code : 25 hours.
We are confident that we can complete the project on-time.
It is important to note that while implementing the class diagram, we were able to identify several potential design patterns that can be used. Our plan is to leverage our extensive class diagram and try to introduce several design patterns as the project progresses. We have most of our classes and interfaces ready on GitHub under "Java" folder.

Next Iteration:

We are planning to start working on our front-end first. This gives us a clear picture of the data that we should store in the database. We have the classes and interfaces ready; however, these might undergo changes in the future. We will build our database and back-end system. Test the back-end. Then we will work on integrating the front-end, back-end and test the code, logic and border cases.