User

+ firstName: string
+ lastName: string
+ userName: string
+ userTag: string (student, alumnus, or staff)
+ nickNames: Set<string>
+ profilePicture: image
+ coverPicture: image
+ pictures: Dict<image>
+ departments: List<Department>
+ socialMedia: Dict<[platform: address]>
+ positions: orderedSet<Experience>
+ graduationYear: int
+ birthday: [year: int, mon: int, day: int]
+ groups: orderedSet<Group>
+ connections: orderedSet<User>
+ connectionRequests: orderedList<[User, message]>
+ outgoingRequests: orderedList<[User, message]>
+ messages: orderedList<chat>
+ posts: list<Post>
+ lastLocation: geoData
+ settings: Dict

CONSTRUCTORS
+ constructor(): User
+ constructor(firstName, lastName, userName, userTag, graduationYear): User

MUTATERS
+ changeName(firstName, lastName): void
+ changeUserName(newUserName): void
+ editNickNames([newNickName]): void
+ correctBirthday(year, mon, date): void
+ updateUserTag(newTag): void
+ changePicture(newPicture, type: "profile" or "cover"): void
+ editPicturesList(command: "remove" or "add", item): void
+ changeDepartment(newDepartment): void
+ addPosition(Position): void
+ editPostitions(command: "add" or "remove", item): void
+ addSocialMedia(platform, address): void
+ addPost(Post): void
+ editPostList(Post): void
+ removePost(Post): void
+ updateLocation(geoData): void
+ updateSettings(Dict): void
+ updateMessages(Chat): void
+ addConnection(User): void
+ sendRequest(User): void

GETTERS:
+ getName(type: "first", "last", "both"): string
+ getUserName(): string
+ getUserTag(): string
+ getNickNames(): List<string>
+ getBirthday(): Array<year: int, month: int, day: int>
+ getProfilePicture(): image
+ getcoverPicture(): image
+ getDepartment(): List<Department>
+ getSocialMedia(platform): string
+ getCurrentPosition(): Position
+ getPositions(): List<Position>
+ getLastLocation(): geoData
+ getSettings(): Dict

UTILITY
+ importFromResume(file): void

POST

+ Text: string
+ owner: User
+ co_owners: List<User>
+ timeOfPost: string ("year:mm:dd hh:mm:ss")
+ timeOfEdit: string ("year:mm:dd hh:mm:ss")
+ editted: bool
+ departments: List<Department>
+ Images: List<image>
+ tags: Set<string>
+ scope: string<string>
+ Likes: Set<User>
+ private: bool
+ lifeUpdate: bool
+ comments: Dict<User: string>
+ acceptComments: bool

CONSTRUCTOR
+ constructor(text, [List<image>], List<tags: strings>): Post
+ constructor(text, [List<image>], List<tags: strings>, privacy: bool, scope: string): Post

MUTATERS
+ editPost(): void
+ addOther(List<User>): void
+ changeDepartment(List<Department>): void
+ editText(newString): void
+ editImages(command: "add" or "remove", item): void
+ changeScope(List<string>): void
+ changeTags(List<string>): void
+ deleteInteractions(index: item): void

+ negatePrivacy(): void
+ negateCommentPermission(): void

GETTERS
+ getOwner(): User
+ getOtherUsers(): List<User>
+ getDepartments(): List<Department>
+ getImages(): List<image>
+ getTags(): Set<string>
+ getScope(): set<string>
+ isPrivate(): bool
+ getLikes(): Set<Users>
+ getComments(): Dict<User: string>
+ isLifeUpdate(): bool
+ acceptComments(): bool

FEED

+ owner: User
+ other_users: List<User>
+ timeOfPost: string ("year:mm:dd hh:mm:ss")
+ scope: string<string>
+ Likes: Set<User>
+ private: bool
+ lifeUpdate: bool
+ comments: Dict<User: string>
+ acceptComments: bool
+ filters: List<Department>, List<Tags>, List<UserInfo>
+startPost: bool
+status: String

CONSTRUCTOR
+ constructor(): Feed
+ constructor(user, other_users, startPost, status, filters): Feed

MUTATERS
+ editPost(): void
+ viewOtherUsers(List<User>): void
+ filterFeed(List<Department>, List<Post>, List<UserInfo>): void
+ changeScope(List<string>): void
+ negatePrivacy(): void
+createStatus(String): void

GETTERS
+ getOwner(): User
+ getOtherUsers(): List<User>
+ getFilter(): List<Department>, List<Tags>, List<UserInfo>
+ getImages(): List<image>
+ getTags(): Set<string>
+ getScope(): set<string>

+ isPrivate(): bool
+ getLikes(): Set<Users>
+ isLifeUpdate(): bool

DEPARTMENT

+ deptName: string
+ deptPosition: string
+ deptPicture: image
+ pictures: Dict<image>
+ alumniList: List<User>
+ staffList: List<User>
+ socialMedia: Dict<[platform: address]>
+ groups: orderedSet<Group>

CONSTRUCTORS
+ constructor(): Department
+ constructor(deptName, deptPosition): Department

MUTATERS
+ changeName(newDeptName): void
+ changePicture(newPicture): void
+ editPicturesList(command: "remove" or "add", item): void
+ addUser(User, type: “alumni” or “staff”): void
+ editAlumniList(command: "add" or "remove" or “get”, item): void
+ editStaffList(command: "add" or "remove" or “get”, item): void
+ editSocialMedia(command: "add" or "remove", platform, address): void

GETTERS:
+ getName(type: "first", "last", "both"): string
+ getPicture(): string
+ getType():string
+ getSocialMedia(platform): string
+ getStaff(User/String): User
+ getAlumni(USer/String): User

UTILITY
+ isAlumni(User/string): bool
+ isStaff(User/string): bool

Position
+ StartDate: String
+ EndDate: string
+ Accolades: List<string>
+ Company: string
+ Description: string

CONSTRUCTORS
+constructor(): Position

+constructor(startDate, endDate, accolades, description)

MUTATERS
+changeStatDate(date): void
+changeEndDate(date): void
+addAccolade(accolade): void
+removeAccolade(accolade): void

GETTERS
+getStartDate(): String
+getEndDate(): string
+getAccolades(): list
+getCompany(): string
+getDescription(): string

UTILITY
+ importFromResume(file): void
