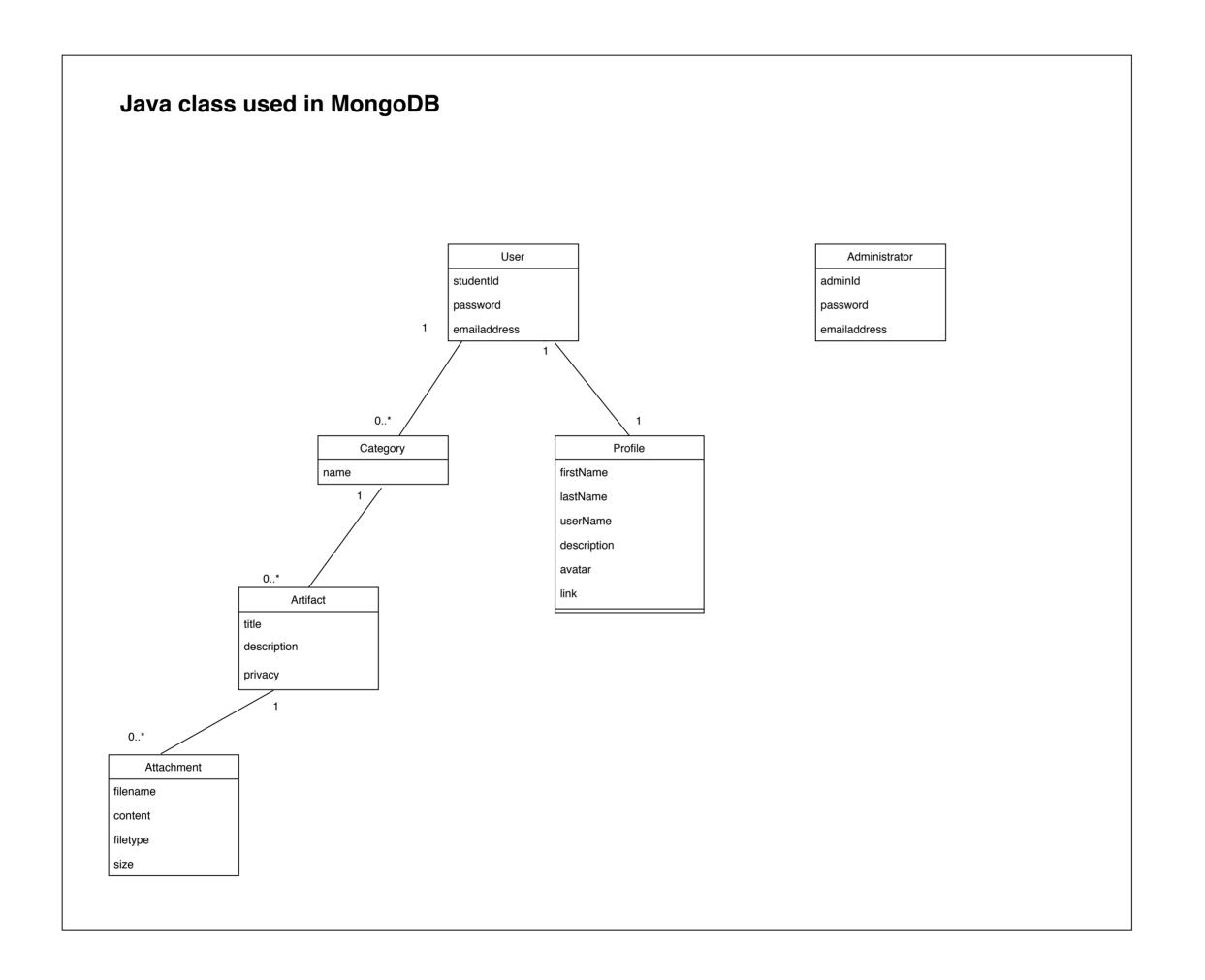
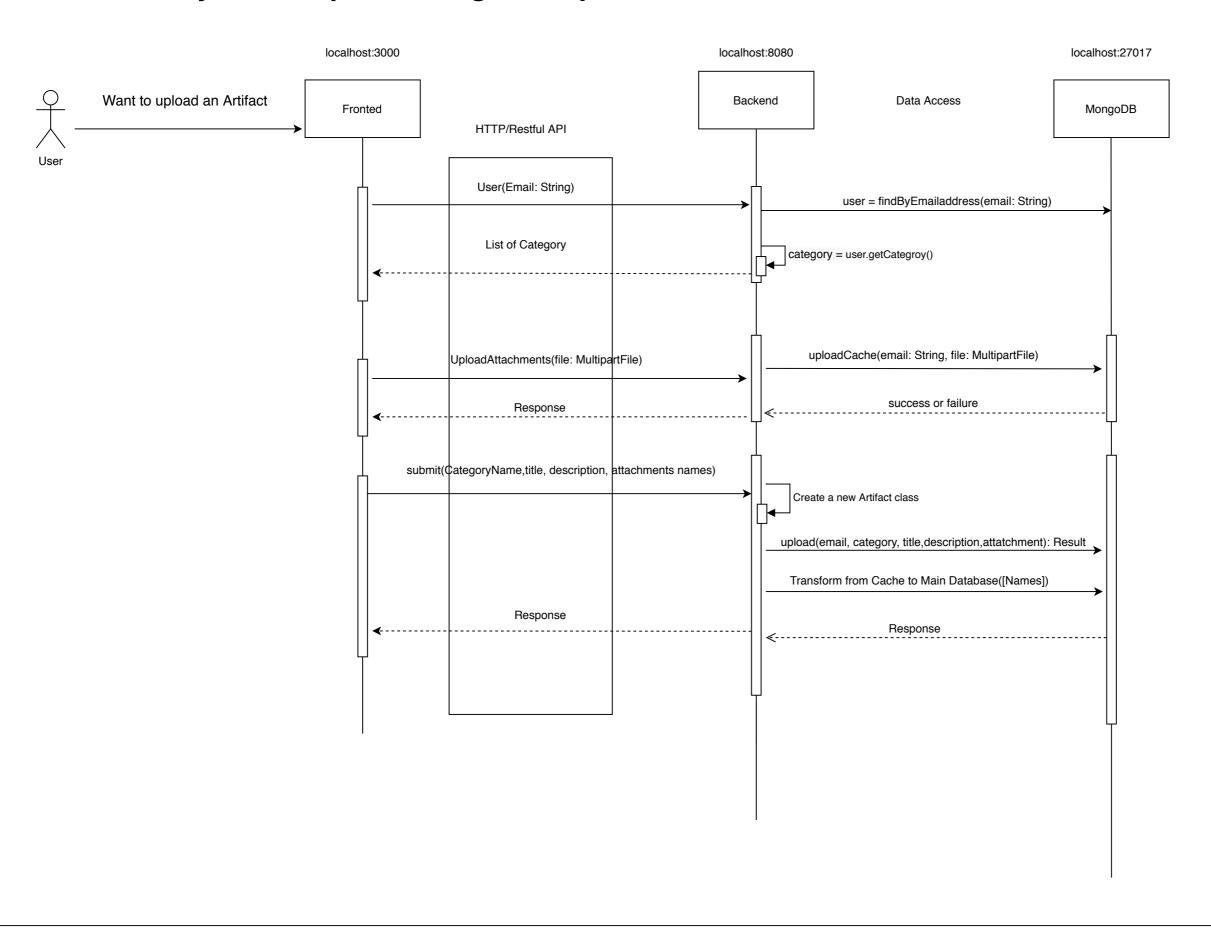
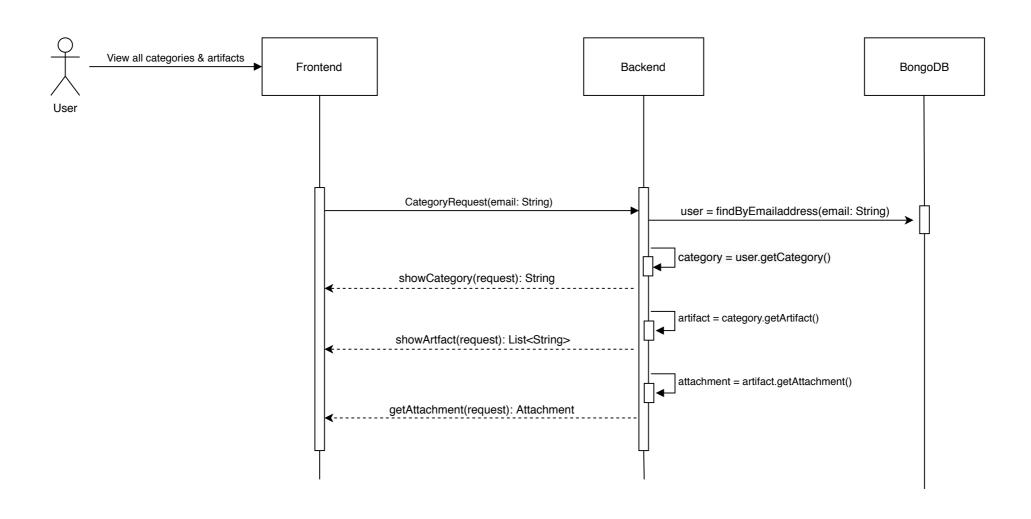
Spring MVC 3-level architecture Note: Every dependency without label here is : Manage Data(Service -> Data Access) Note: Each class in this architecture is a sinleton and the property is set automatically by Spring framework Data Service Layer Access Controller Layer AccountController AccountService Uses + getAllUsers(): List<User> + getAllUsers(): List<User> register(request:RegisterRequest): Result register(request:RegisterRequest): Result + registerAdmin(adminId:String, email:String, password:String): Result + login(email:String, password:String): Result registerAdmin(adminId:String, email:String, password:String): Result login(email:String, password:String): Result UserRepository + findByEmailaddress(emailaddress:String): User + findByStudentId(studentId:String):User **ArtifactService** + findUserByFirstname(firstname:String): List<User> ArtifactController + findUserByLastname(lastname:String): List<User> + showCategoryArtifactAttachments(): Map<String, Map<String, List<String>>> + findUserByUsername(username:String): List<User> + showCategoryArtifactAttachments(): Map<String, Map<String, List<String>>> + upload(email:String, category:String, title:String, description:String, + findUserByLink(link:String): User + upload(email:String, category:String, title:String, description:String, attachment:List<String>, privacy:String): Result Uses attachment:List<String>, privacy:String): Result +deleteArtifact(request: ViewArtifact): Result +deleteArtifact(request: ViewArtifact): Result +viewAttachment(request:ViewAttachmentRequest): ViewAttachmentResult +viewAttachment(request:ViewAttachmentRequest): ViewAttachmentResult +switchPrivacy(email:String, category:String, artifact:String, privacy:String): Result +switchPrivacy(email:String, category:String, artifact:String, privacy:String): Result CacheService CacheController Uses uploadCache(email:String, file:MultipartFile): Result CacheRepository - uploadCache(email:String, file:MultipartFile): Result + clearCache(email:String):Result - clearCache(email:String):Result + findByEmailaddress(emailaddress:String): UserCache + deleteByEmailaddress(emailaddress:String): Void + findAllByEmailaddress(emailaddress:String): List<UserCache> CategoryController CategoryService + addCategory(request:CategoryRequest): Result - addCategory(request:CategoryRequest): Result Uses + deleteCategory(request:CategoryRequest): Result - deleteCategory(request:CategoryRequest): Result + showCategory(request:AllCategoryRequest): CategoryRequest + showCategory(request:AllCategoryRequest): CategoryRequest GuestService GuestController Uses -findPubByLink(link:String): Map<String, Map<String, List<String>>> + findPubByLink(link:String): Map<String, Map<String, List<String>>> + findProfileByLink(link:String): Profile + findProfileByLink(link:String): Profile AdminRepository + findByEmailaddress(emailaddress:String): Administrator **ProfileController ProfileController** + findAllByAdminId(adminId:String): Administrator Uses updateAvatar(email:String, image:MultipartFile): Result + updateAvatar(email:String, image:MultipartFile): Result + deleteAvatar(email:String): Result + deleteAvatar(email:String): Result + getProfile(email:String): Profile + getProfile(email:String): Profile + updateProfile(email:String, firstname:String, lastname:String/username:String): Result + updateProfile(email:String, firstname:String, lastname:String, username:String): Result + generateLink(email:String): String + generateLink(email:String): String SearchService SearchController Uses + showUser(info: String): List<User> + showUser(info: String): List<User>



System sequence diagram: Upload an artifact

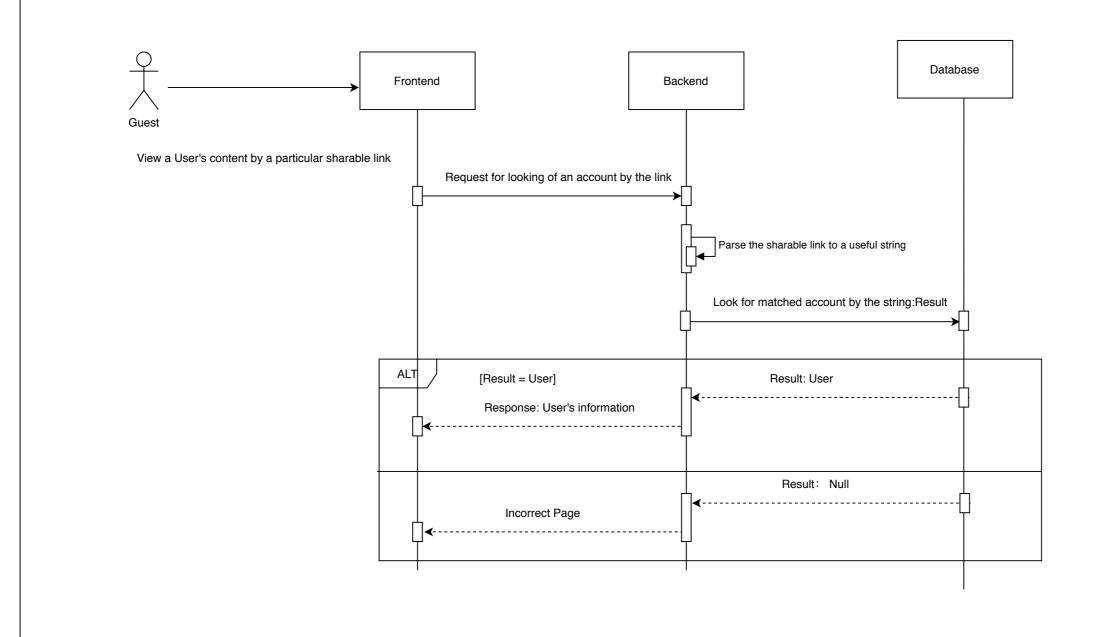


System sequence diagram: View all artifacts as Owner/Guest

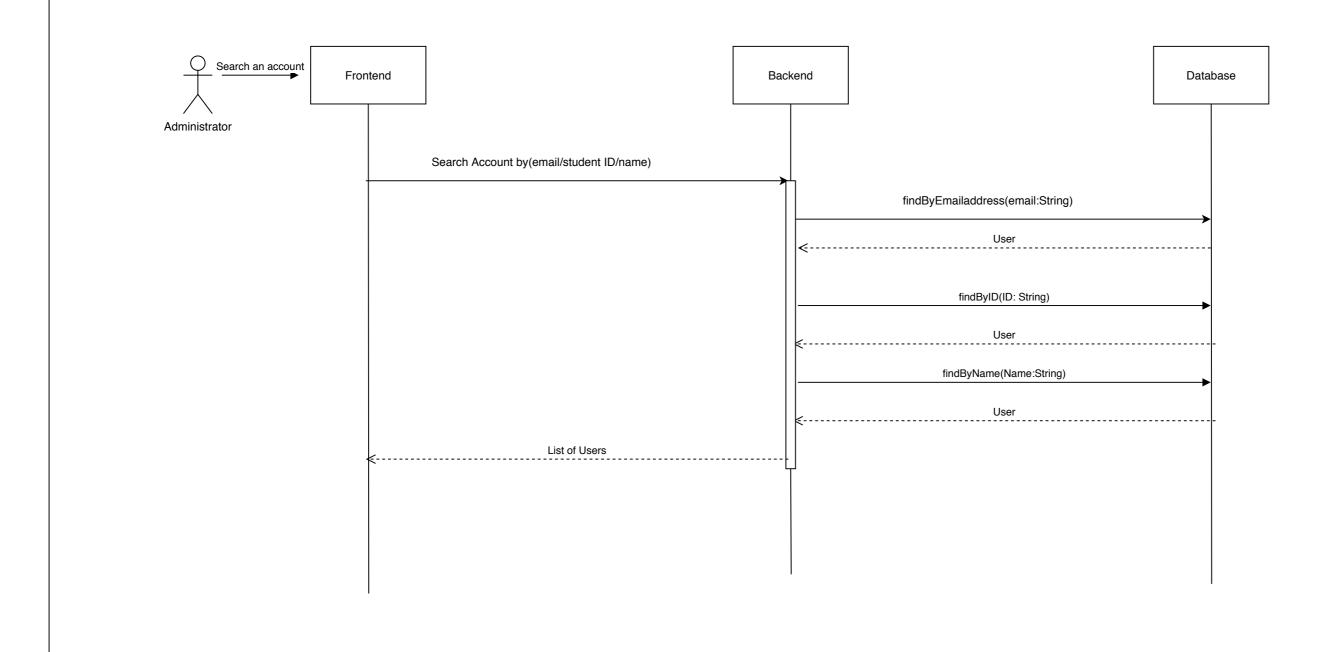


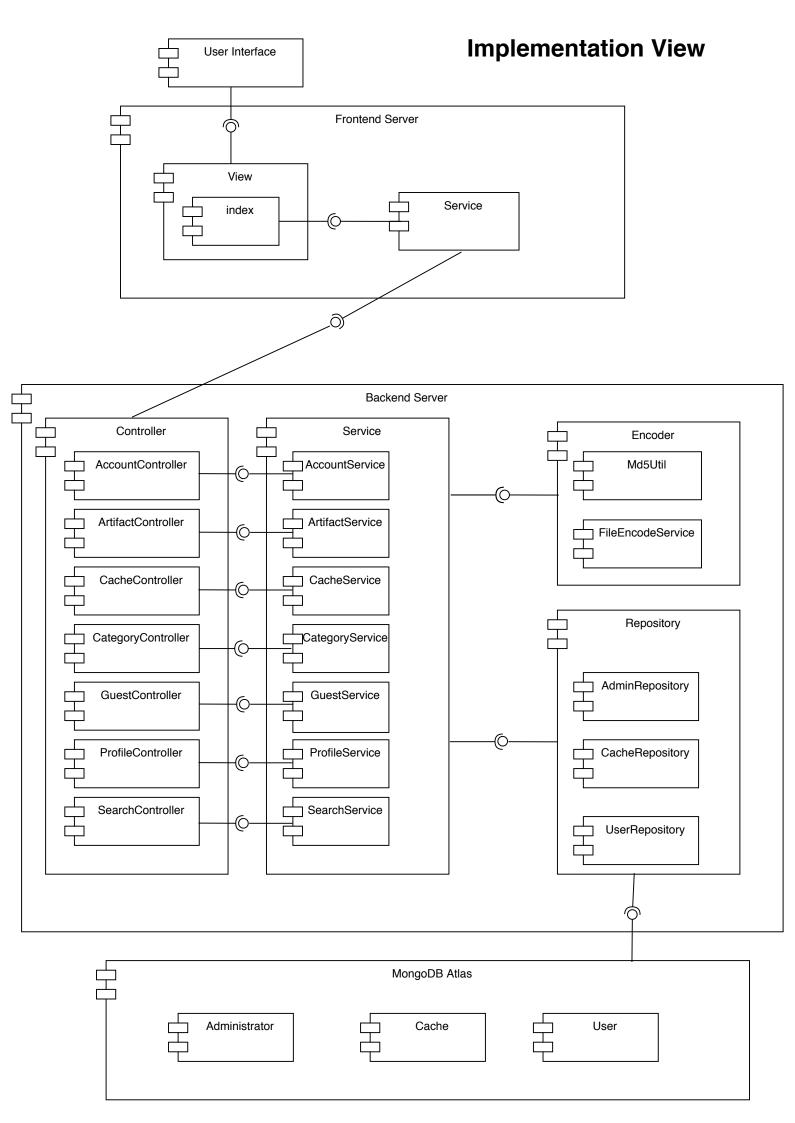
System sequence diagram: User generates a new sharable link generate a new sharable link Frontend Backend Database Student Request for generating a new sharable link findByEmailaddress (email: String) Generate a new hash string(Email,ID,Time) Save hash string in User of database Show the latest link to the user

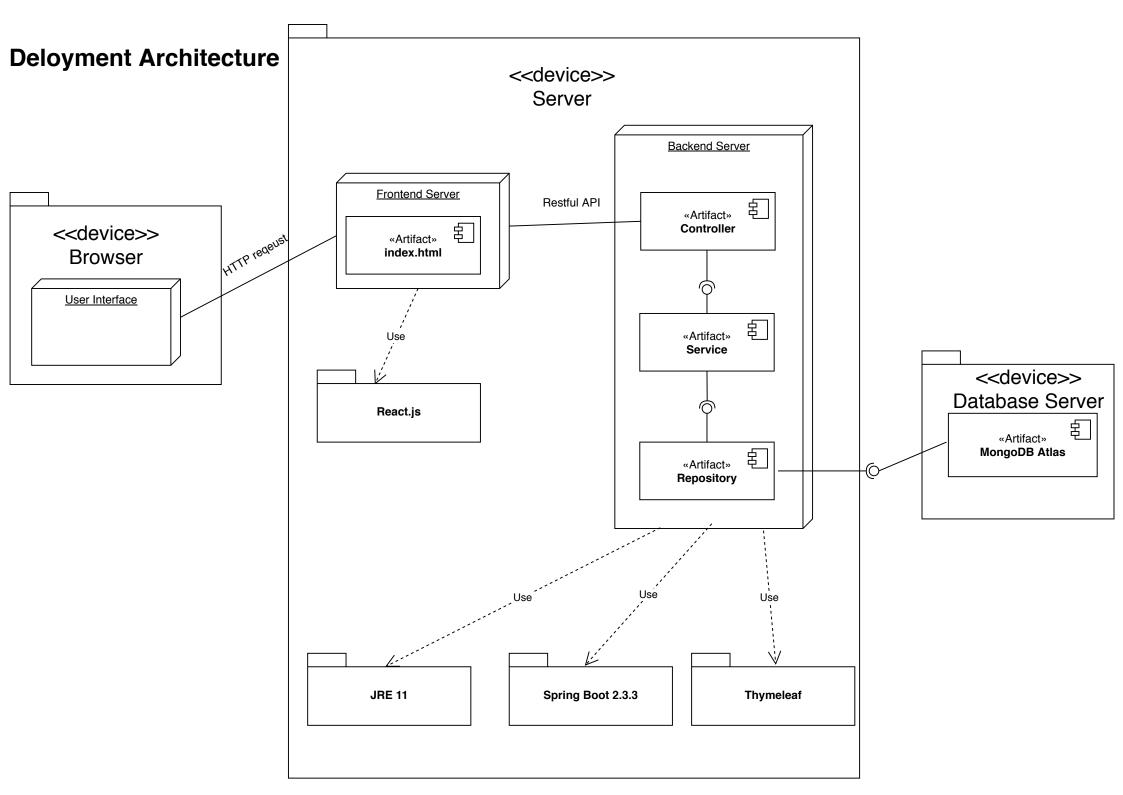
System sequence diagram: Guest view content through a sharable link



System sequence diagram: Administrator search a particular account from the system







Scenarios

Use Case 1: Uploading File

Scope: ePortfolio system

Primary Actor: Student

Supporting actor: System Administrator

Offstage actor: Database

Preconditions:

Student has registered an account in the database.

Success guarantee:

Network is stable and no file is lost.

Students get verified by the system successfully.

Main successful scenarios:

- 1. Student has used the correct email and password to log in the system.
- 2. Student clicks on Artifact.
- 3. Student clicks or drags files to upload.
- 4. Files will be first loaded in a cache space to detect malicious attacks.
- 5. System returns success to the student.
- 6. The student chooses or creates a new category for this artifact.
- 7. Student writes title, description and chooses privacy of the artifact.
- 8. Student clicks upload to confirm creating the artifact.
- 9. System redirects the student to his homepage.

Alternate Scenarios:

- 1. Fail to login
 - a. Email is invalid
 - b. Incorrect password
- 2. Same category name/artifact name

Use Case 2: Guest view

Scope: ePortfolio system

Primary Actor: Guest

Supporting actor: Student

Offstage actor: Database

Preconditions:

Guests have the correct and usable links from students.

Success guarantee:

Network is stable and no file is lost.

Main successful scenarios:

- 1. Guest enters the correct link in the browser.
- 2. System will route the webpages and return the public content of the students.
- 3. Guests can click on the Home tag to view the profile of the student.
- 4. Guests can click on the Category tag to view the artifacts of the student.

Alternate Scenarios:

- 1. The link has already been discarded
 - a. User shares link with the guest
 - b. User generate a new link and discard the old one
 - c. Guest uses the old link to view the content of the user
 - d. An error page will be returned to the guest until he gets the latest link