# Háskóli Íslands

## Software project 1
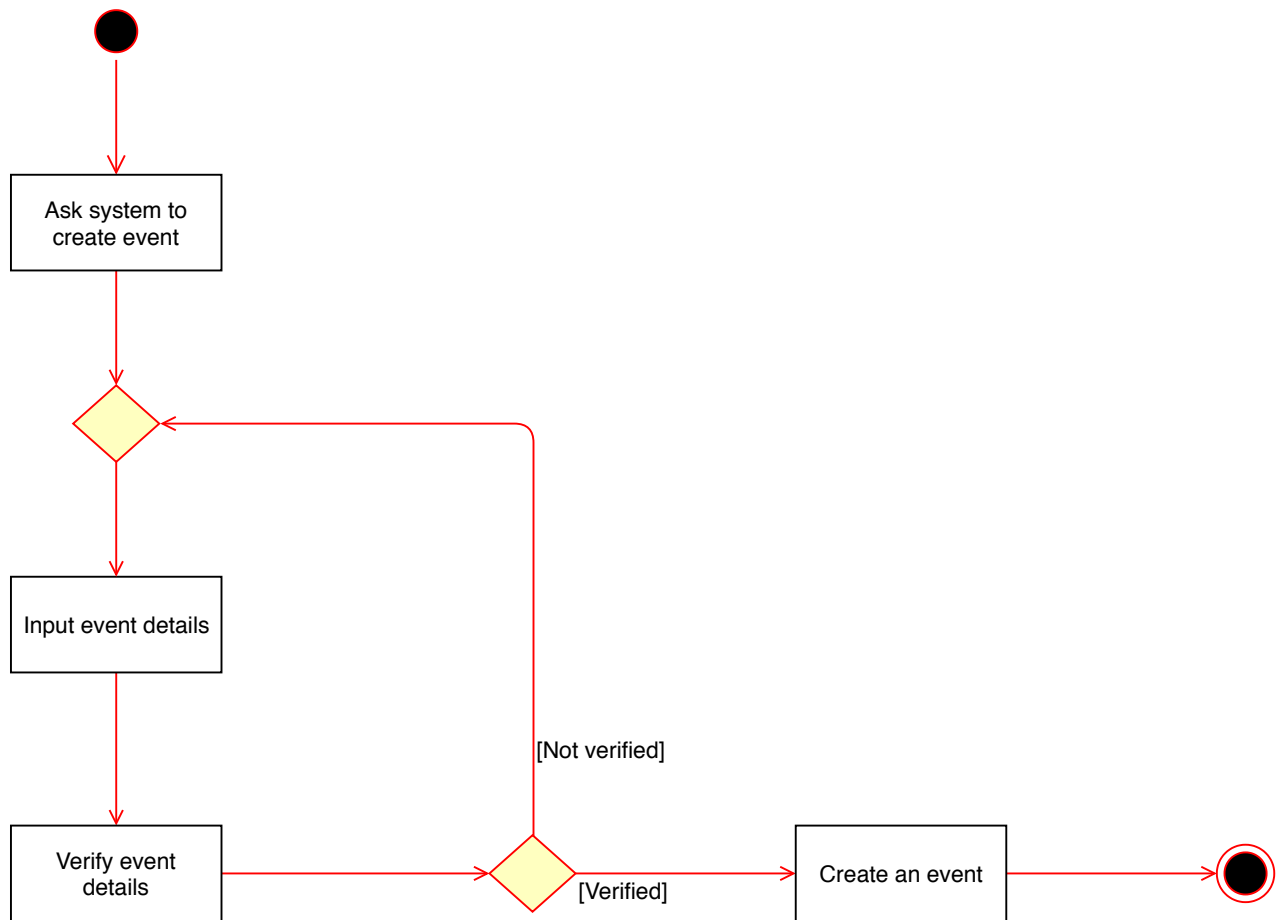
# Project Plan and Requirements

**Team 29**

| | |
|---|---|
| Atli Marcher Pálsson | *amp17@hi.is* |
| Haukur Þór Ágústsson | *htha13@hi.is* |
| Pétur Sigurðsson | *pes14@hi.is* |
| Jón Rúnar Baldvinsson | *jrb6@hi.is* |

# Create Event

Ask system to create event

Input event details

Verify event details

[Not verified]

[Verified]

Create an event

# Register / Login



[Existing user]

[New user]

Ask system to create an account

Enter account details

Verify account details

[Not verified]

[Verified]

Create new account

[Successful]

[Failed]

Attempt login

Input user credentials

2

**@Controller EventController**

- eventService: EventService
- userService: UserService

+ EventController(eventService: EventService, userService: UserService)
+ getEvents(startDate: Date, endDate: Date, token: String) List<Event>
+ findOneEvent(id: String) Event
+ postEvent(event: Event, token: String): Event
+ updateUserList(usernames: List<String>, id: String, token: String): Event
+ updateEvent(event: Event, id: String, token: String) Event
+ deteteEvent(id: String, token: String) Event

**@Entity Event**

- id: Long
- startDate: Date
- endDate: Date
- title: String
- description: String
- users: List<User>

+ Event()
+ Event(startDate: Date, endDate: Date,
  title: String, description: String)
[ Getters & Setters ]

**<<Interface>> EventService**

+ save(user: User, event: Event): Event
+ delete(user: User, id: Long)
+ findByDate(user: User, startDate: Date, endDate: Date): List<Event>
+ findOne(id: Long): Event
+ updateEvent(user: User, id: Long, event: Event) Event
+ updateUserList(user: User, id: Long, usernames: List<String>): Event

**@Service EventServiceImplementation**

- repository: EventRepository
- userRepository: UserRepository

+ EventServiceImplementation(eventRepository: EventRepository, userRepository: UserRepository)
+ save(user: User, event: Event): Event
+ delete(user: User, id: Long)
+ findByDate(user: User, startDate: Date, endDate: Date): List<Event>
+ findOne(id: Long): Event
+ updateEvent(user: User, id: Long, event: Event): Event
+ updateUserList(user: User)

**<<Interface>> EventRepository**

+ save(event: Event): Event
+ findByStartDateBetween(startDate: Date,endDate: Date): List<Event>
+ findOneById(id: Long): Event
+ delete(event: Event)

**@Entity User**

- id: Long
- name: String
- username: String
- password: String
- token: String
- events: List<Event>

+ User()
+ User(name: String, username: String,
  password: String)
[ Getters & Setters ]

**@Controller LoginController**

- userService: UserService

+ LoginController(userService: UserService)
+ loginUser(user: User): User
+ registerUser(user: User): User

**@Controller UserController**

- userService: UserService

+ UserController(userService: UserService)
+ registerNewUser(user: User) User
+ deleteExisting(user: User)
+ findUser(id: Long): User
+ updateExisting(user: User) User
+ loginUser(user: User) User

**@Service UserServiceImplementation**

- repository: UserRepository
- passwordEncoder: BcryptPasswordEncoder

+ UserServiceImplementation(repository: UserRepository)
+ save(user: User): User
+ delete(user: User)
+ findOne(id: Long): User
+ update(user: User): User
+ login(user: User): User
+ findByToken(token: String): User
- cleanUser(user: User): User
- cleanUserList(userList: List<User>): List<User>

**<<Interface>> UserService**

+ save(user: User): User
+ delete(user: User)
+ findOne(id: Long): User
+ update(user: User): User
+ login(user: User): User
+ findAll(): List<User>
+ findByToken(token: String): User

**<<Interface>> UserRepository**

+ findUserByUsername(username: String): User
+ findAll(): List<User>
+ findOneById(id: Long): User
+ findByToken(token: String): User

4

## Connection

- instance: Connection
- URL: String
- JSON: MediaType

---

- Connection()
+ getInstance(): Connection
- post(endPoint: String, obj: Object, c: Callback): void
- get(endPoint: String, c: Callback): void
- post(endPoint: String, obj: Object, token: String, c: Callback): void
- get(endPoint: String, token: String, c: Callback): void
+ loginUser(user: User, c: Callback): void
+ registerUser(user: User, c: Callback): void
+ createEvent(event: Event, token: String, c: Callback): void
+ editEvent(event: Event, token: String, c: Callback): void
+ deleteEvent(event: Event, token: String, c: Callback): void
+ getEvents(startDate: Date, endDate: Date, token: String, c: Callback): void
+ getEventByID(id: Long, token: String, c: Callback): void
+ shareEvent(users: List<String>, event: Event, token: String, c: Callback): void
+ deleteUser(token: String, c: Callback): void

## MainActivity

- connection: Connection

## TokenStore

- PREF_NAME: String
- KEY: String

---

+ getToken(context: Context): String
+ setToken(token: String, context: Context): void
+ deleteToken(context: Context): void

## LoginActivity

- createNew: boolean
- username: String
- password: String
- name: String
- c: Connection

---

- finishLogin(): void
- login(user: User): void
- create(user: User): void
- sendResult(): void
- setToken(): void

## User

- id: Long
- name: String
- username: String
- password: String
- token: String
- events: List<Event>

---

+ User()
+ User(name: String, username: String,
    password: String, token: String)
[ Getters & Setters ]

+ Event(id: Long, startDate: Date, endDat
    title: String, description: String)
[ Getters & Setters ]

## EventActivity

- c: Connection
- event: Event
- type : String

---

- share() : Void
- addUser(user : User) : Void
- createEvent() : Void
- editEvent() : Void

## Event

- id: Long
- startDate: Date
- endDate: Date
- title: String
- description: String
- users: List<User>

---

+ Event()
+ Event(id: Long, startDate: Date, endDat
    title: String, description: String)
[ Getters & Setters ]