Hugbúnaðarverkefni 1

1)

1.5 Vision Statement

The essence of this program will be to provide a functional and secure clock-in program for construction companies and contractors, that tracks and stores employee work hours in a format suitable for employers to review.

The worksite clock in program is meant to function as a secure employee shift manager on worksites where multiple independent
companies/contractors can collaborate together. It will do so by marking employee clock-in data with a tag representing the company they associate with.

The clock-in data is to be stored securely with encryption in an online server unlike manual records or single company solutions, further enabling collaborations and cohesion between companies.

1.2 Business Opportunity

Fragmented or manual time tracking on multi-contractor worksites often leads to disputes, errors and wasted time. This system eliminates those problems by implementing a more unified, robust and auditable system. Representatives of the contracted companies will be able to independently check the data of their employees, additionally site managers can access the data of all employees.

The program will work to efficiently collect data and allow companies to ensure proper working order, such as allowing contractors to check the status of their employees, and allowing site managing companies to ensure that the working hours of the contracted companies/employees are adhered to, alongside tracking total work hours for each given project. As construction projects grow in size and regulatory requirements and pressure increase, the demand for a system such as this will increase.

2.2 Scope of Initial Release

The initial release of the program will focus on local side functionality and proper storing of data as described above. Given additional time server side functionality alongside secure encryption of said data can be implemented.

2)

2.1

2.3.1 Use case name

Reconfigure and edit employee shifts [UC1]

2.3.4 Primary actor

Shift Manager

2.3.6 Preconditions

Manager is logged in to his user account and authenticated as a manager under target company

2.3.7 Success guarantee

Reconfigured shift is updated accordingly and saved

Target employee's shift timeline is conflict free and persisted

All edits are versioned with a reason note and the editor identity

2.3.8 Main success scenario

1. Manager opens a "Shifts" overview in system for a date range, employee, or location
2. Manager is presented a list of shifts grouped by day with statuses such as "Ongoing", "Finished", "Overlap" and "Edit requested"
3. Manager selects a shift and decides to edit it
4. Manager adjusts start/end time, task, location and adds a reason
5. System validates (no negative duration, no overlaps)
6. Manager saves
7. System writes change, creates audit entry, recalculates totals
8. Overview shows conflict-free timeline and updated totals

2.3.9 Extensions / alternate scenarios

2-6) Employee request to edit shift exists

1. Manager opens request
2. Manager analyzes and changes the requested changes and approves them/rejects
3. Manager adds a note reasoning why

2-7) Manager sees ongoing shift

1. Manager options for "End now"
2. System sets clock-out time as current time

5) Invalid time

1. Validation error, no save
2. Manager is presented an error message

5-6) Overlap detected

1. Manager is presented with options to combat overlap, such as trim current or existing shift, cancel
2. Manager decides on an option and saves

2.1.13 Open issues

Should editing always create a new shift or only edit currently existing - or both?

How much time before automatically ending a shift - should they even automatically end?

2.2

2.2.1 Use case name

Manage Tasks [UC2]

2.2.4 Primary actor

Company manager

2.2.6 Preconditions

Manager is logged in on his user account and authenticated as a manager under target company

2.2.7 Success guarantee

Added/Changed/Removed task is saved correctly to persistent storage

Shifts relating to changed task are updated accordingly

Employees are notified of the updated availability

Audit logs record exact changes with actor identity

2.2.8 Main success scenario

1. Manager is presented an overview of company assets
2. Manager decides to add a new task
3. Manager selects under which location, if any, the task should be
4. Name and description are entered
5. Manager saves the task
6. System verifies the request (no blank name and not too long/short)
7. System saves the task, along with the audit log
8. Task is presented in overview of company assets and is visible to employees

2.2.9 Extensions / alternate scenarios

2-4) Edit task

1. Manager decides to edit existing task
2. Edits existing name/description/is finished

2-7) Delete task

1. Manager decides to delete existing task
2. System verifies no active shifts depend on it/changes shifts accordingly
3. System deletes the task and logs with an audit log

6-4) Name size invalid/is blank

1. Validation error, system rejects
2. Manager is presented an error message

6) Collateral on shifts

1. System responds with shifts that would be affected in what way
2. Presents situation to manager
3. Manager decides on an option, update all shifts, delete shifts, update tasks of ongoing shifts, cancel

2.2.13 Open issues

Should deleted tasks also delete shifts relating to them?

Should tasks only be soft deleted? (marking them as deleted) And in that case, should they be un-deleteable or visible at all?

Can tasks be edited with ongoing shifts working on them?

2.3

2.3.1 Use case name

Clock-in/Clock-out [UC3]

2.3.4 Primary actor

Employee

2.3.6 Preconditions

Employee is logged in to his user account and authenticated as an employee under target company

2.3.7 Success guarantee

Ongoing/finished shift gets correctly saved to persistent storage

All shifts are within reasonable bounds of length

2.3.8 Main success scenario

1. Employee arrives at work
2. Selects location if so is needed and task he intends to work on
3. Clocks in through system
4. System validates that there is no currently ongoing shift for target employee
5. System saves the new shift as ongoing
6. Employee clocks out at the end of the day
7. System validates that the shift is not too long/short etc.
8. System saves that shift as completed

2.3.9 Extensions / alternate scenarios

5a) Employee decides to change task

1. Selects new task
2. System saves old task time
3. System starts new task at current time

5b) Employee decides to change location

1. Selects new location
2. Selects new task if the old task was location-specific
3. System saves the old location/task time
4. System starts the new location/task at current time

6-8) Employee forgets to clock out

1. System automatically ends shift after n hours
2. Adds a note to the task identifying it as having been automatically ended

2.3.13 Open issues

For breaks, should employee be forced to clock out then in again, or have a separate function to add a break?

Is it necessary to select a task to begin a shift?

2.4

Brief use cases:

- [UC4] Overview of currently ongoing shifts

A manager authenticates self through the system and wants to see which employees are currently working. He navigates the system to an overview of currently ongoing shifts, and the system presents a list of shifts that are currently ongoing. The manager takes note of all the entries and performs necessary actions to keep the company in good health, then leaves the overview satisfied.

- [UC5] CRUD operations on locations

A manager authenticates self through the system and navigates to the company he intends to operate on. He wants to adjust the locations available at the company. Using the system, he navigates to an overview of the company's locations and performs CRUD operations to the company's benefit. For each operation, he saves and the system records the changes. The manager then leaves the overview.

- [UC6] Request an edit to a shift

An employee authenticates self through the system and navigates to the company where he intends to work. The system notifies him that he forgot to clock out of the last shift. Through the system, he constructs a request to his managers to change his previous shift to what he considers fair. The system records the request and presents it to the managers, where they can approve/reject any requests the employee made. Once the managers have decided on a judgement, the system records the changes and the employee receives a message from the system, notifying him of the result.

- [UC7] Contracted under multiple companies

A user works two part-time jobs at two different companies. At the start of the day, he uses the system to clock in to the first job and works half the day there, until he clocks out of that company. He then returns to an overview of the companies he is contracted under, and opts to enter the company where his next job is. There, he can clock in and work the rest of the day and the system records his two shifts that day as under each corresponding company.

Removed UC8 since it requires functionality outside the scope of the project, and doesnt align with the vision for the program.

- [UC8] Overview of a company’s status in statistical format over time period

A manager that is authenticated as such under the target company, wants to get a clear overview of how much his employees worked this month in a common statistical format. He navigates the system to get to an option to export the statistics of all employees and tasks from any date range. He adjusts the date range to be one month and asks the system to export. The system presents him with a file in a commonly used statistical format that is available for download. The manager downloads the file and can now use it for any purpose the company desires.

- [UC9] Calendar-like overview of shift history

An authenticated user is curious about his work history and uses the system to get a calendar-like overview of his shift history. This includes all companies he is contracted under. He then opts to only show a specific company and the system responds. The system shows which days he worked and what time and any other relevant info the user desires. The user internalizes this information and leaves the overview satisfied.

- [UC10] Change history of assets under company

An authenticated user is identified as a manager under the target company, and uses system to see the audit history of an asset under his company, be it a location, task or shift, or a user contract. The system reads this information and presents it to the user. Visible there, is what asset is being inspected, what changed at what time and by who. The manager now knows the history of any asset of his liking and leaves.

- [UC11] User registration and login

A user wants to get access to the program and opts to register for the service, creating an account that includes a personal identifier to him, allocating a password to his account. The program saves this information to persistent storage, obfuscating the password. The user then logs in to the service using his identifier and password and gets access to further operations.

3)

Time estimation calculated in hours.
The lower the priority number, the more vital the use case is, P1 > P2 > P3

Use Case	Time estimation	Priority
UC3	8	1
UC1	20	2
UC2	25	3
UC4	6	4
UC5	5	5
UC6	4	6
UC7	9	8
UC8	8	9
UC9	7	7
UC10	6	10

4)

Week	Use Cases	Expected hours	P.O.	Sprint	Consultation
1	UC3	5	Ó.S.S.	1	**A1 pres.**
2	UC3, UC1, UC4	3, 3, 2	Ó.S.S.	1	Model Draft
3	UC1, UC4, UC6	6, 4, 2	E.A.E	2	**A2 pres.**
4	UC1, UC5, UC6	6, 2, 2	E.A.E	2	Dev support
5	UC1, UC5	5, 3	R.B.I	3	Dev support
6	UC2, UC9, UC11	5, 3	R.B.I	3	**A3 pres.**
7	UC2, UC7, UC9	5, 5, 4	R.B.I	3	Techn. fixes
8	UC2, UC7	5, 4	D.S.H	4	Testing
9	UC2, UC8, UC10	5, 4, 3	D.S.H	4	finalization
10	UC2, UC8, UC10	5, 4, 3	D.S.H	4	**A4 pres placeholder**

5)

https://github.com/Amaragance682/hugbunadarverkefni1