**MINISTRY OF EDUCATION AND TRAINING**

**FPT UNIVERSITY**

Capstone Project Document

**Babysitter In Demand**

|  |  |
| --- | --- |
| **Group 13 - IS** | |
| **Group members** | Phùng Thiên Phúc – SE62639 (Leader)  Phạm Hải Dương – SE62746  Hồ Tấn Kỳ – SE61974  Hoàng Nhật Đông – SE62357 |
| **Supervisor** | Kiều Trọng Khánh |
| **Ext. Supervisor** | N/A |
| **Project Code** | 1. BSD |

– Ho Chi Minh City, 10 September 2019 –

List of Contents

[A. Outline 8](#_Toc27003264)

[1. Project Information 8](#_Toc27003265)

[2. Introduction 8](#_Toc27003266)

[3. Current Situation 8](#_Toc27003267)

[4. Problem Definition 8](#_Toc27003268)

[5. Proposed Solution 9](#_Toc27003269)

[5.1. Feature functions 9](#_Toc27003270)

[5.2. Values and challenges 10](#_Toc27003271)

[6. Functional Requirements 10](#_Toc27003272)

[1.1. Staff Component 10](#_Toc27003273)

[1.2. Parent Component 10](#_Toc27003274)

[1.3. Babysitter component 11](#_Toc27003275)

[1.4. Authentication component 11](#_Toc27003276)

[1.5. Recommendation component 11](#_Toc27003277)

[7. Roles and Responsibilities 12](#_Toc27003278)

[B. Software Project Management Plan 12](#_Toc27003279)

[1. Problem Definition 12](#_Toc27003280)

[1.1. Name of this Capstone Project 12](#_Toc27003281)

[1.2. Problem Abstract 12](#_Toc27003282)

[1.3. Project Overview 12](#_Toc27003283)

[2. Project organization 14](#_Toc27003284)

[2.1. Software Process Model 14](#_Toc27003285)

[3. Project Management Plan 17](#_Toc27003286)

[3.1. Software development life cycle 17](#_Toc27003287)

[3.2. Phase Detail 19](#_Toc27003288)

[3.3. Task sheet 21](#_Toc27003289)

[3.4. All Meeting Minutes 21](#_Toc27003290)

[4. Coding Convention 22](#_Toc27003291)

[C. Software Requirement Specification 22](#_Toc27003292)

[1. User Requirement Specification 22](#_Toc27003293)

[1.1. Guest requirement 22](#_Toc27003294)

[1.2. Parent requirement 23](#_Toc27003295)

[1.3. Babysitter requirement 23](#_Toc27003296)

[1.4. Staff requirement 23](#_Toc27003297)

[1.5. System handler requirement 23](#_Toc27003298)

[1.6. Authorized User 23](#_Toc27003299)

[1.7. Payment 24](#_Toc27003300)

[2. System Requirement Specification 25](#_Toc27003301)

[2.1. System Overview Use Case 25](#_Toc27003302)

[2.2. List of Use Case 26](#_Toc27003303)

[3. Conceptual Diagram 86](#_Toc27003304)

[D. Software Design Description 87](#_Toc27003305)

[1. Design Overview 87](#_Toc27003306)

[2. System Architecture Design 87](#_Toc27003307)

[3. Component Diagram 88](#_Toc27003308)

[4. Detailed Description 90](#_Toc27003309)

[4.1. Class Diagram 90](#_Toc27003310)

[4.2. Interactive Diagram 92](#_Toc27003311)

[5. Database Design 99](#_Toc27003312)

[5.1. Entity relationship diagram(ERD) 99](#_Toc27003313)

[E. System Implementation & Test 101](#_Toc27003314)

[1. Database relationship Diagram 101](#_Toc27003315)

[1.1. Physical Diagram 101](#_Toc27003316)

[2. Framework, technology 102](#_Toc27003317)

[3. Mobile Framework: React Native 104](#_Toc27003318)

[F. References 105](#_Toc27003319)

List of Figures

[Figure 1 - Atern Development Model – DSDM (1) 15](#_Toc27003320)

[Figure 2 – Software development life cycle[1] 17](https://d.docs.live.net/48d30ef320bd90b4/Capstone%20Project/CD/BSD_Document_50pages-1.docx#_Toc27003321)

[Figure 3 – Overview Use Case 25](#_Toc27003322)

[Figure 4 - <Guest> Login 26](#_Toc27003323)

[Figure 5 - <Babysitter> Overview Use Case 29](#_Toc27003324)

[Figure 6 - <Babysitter> Get profile 29](#_Toc27003325)

[Figure 7 - <Babysitter> Edit sitting preferences 31](#_Toc27003326)

[Figure 8 - <Babysitter> Answer invitation 33](#_Toc27003327)

[Figure 9 - <Babysitter>Get invitation 35](#_Toc27003328)

[Figure 10 - <Babysitter> Get invitation list 37](#_Toc27003329)

[Figure 11 - <Babysitter> Get sitting history 39](#_Toc27003330)

[Figure 12 - <Parent> Overview Use Case 42](#_Toc27003331)

[Figure 13 - <Parent> Create babysitting request 43](#_Toc27003332)

[Figure 14 - <Parent> Get babysitting request list 47](#_Toc27003333)

[Figure 15 - <Parent> Get babysitting request 49](#_Toc27003334)

[Figure 16 - <Parent> Accept babysitter 51](#_Toc27003335)

[Figure 17 - <Parent> Add people to trust circle 54](#_Toc27003336)

[Figure 18 - <Parent> Get request history 56](#_Toc27003337)

[Figure 19 - <Parent> Get profile 58](#_Toc27003338)

[Figure 20 - <Payment> Pay 61](#_Toc27003339)

[Figure 21 - <Payment> Refund 63](#_Toc27003340)

[Figure 22 - <System Handler> Overview Use Case 65](#_Toc27003341)

[Figure 23 - <System Handler > Generate identification tokens 65](#_Toc27003342)

[Figure 24 - <System Handler> Save transaction record 68](#_Toc27003343)

[Figure 25 - <System Handler> Matching parent and babysitter 70](#_Toc27003344)

[Figure 26 - <System Handler> Recommend babysitter to parent 73](#_Toc27003345)

[Figure 27 - <Staff> Overview Use Case 75](#_Toc27003346)

[Figure 28 - <Staff> Create user account 76](#_Toc27003347)

[Figure 29 - <Staff> Update user information 79](#_Toc27003348)

[Figure 30 - <Staff> Ban user 81](#_Toc27003349)

[Figure 31 - <Staff> Edit configuration 83](https://d.docs.live.net/48d30ef320bd90b4/Capstone%20Project/CD/BSD_Document_50pages-1.docx#_Toc27003350)

[Figure 32 - Conceptual diagram 86](#_Toc27003351)

[Figure 33 – System architecture design 87](#_Toc27003352)

[Figure 34 – Front-end component diagram 88](#_Toc27003353)

[Figure 35 - Back-end component diagram 89](#_Toc27003354)

[Figure 36 – Class diagram(3) 90](#_Toc27003355)

[Figure 37 - <Guest> Login 92](#_Toc27003356)

[Figure 38 - <Babysitter> Answer invitation 93](#_Toc27003357)

[Figure 39: <Parent> Create babysitting request 94](#_Toc27003358)

[Figure 40 - <Parent> Accept babysitter 95](#_Toc27003359)

[Figure 41 - <Parent> Send feedback 96](#_Toc27003360)

[Figure 42 - <Payment> Pay 97](#_Toc27003361)

[Figure 43 - <System Handler> Match parent and babysitter 98](#_Toc27003362)

[Figure 44 - Entity relationship diagram 99](#_Toc27003363)

[Figure 45 – Physical Diagram 101](#_Toc27003364)

[Figure 46 – NodeJS server(2) 103](#_Toc27003365)

[Figure 47 – Web. Socket communication with browser(4) 103](#_Toc27003366)

List of Tables

[Table 1 - Roles and Responsibilities 12](#_Toc27003367)

[Table 2 - Key responsibilities 16](#_Toc27003368)

[Table 3 – Phase overview 18](#_Toc27003369)

[Table 4 - Phase 1: Requirement Definition 19](#_Toc27003370)

[Table 5 – Phase 2: System and Software Design 20](#_Toc27003371)

[Table 6 – Phase 3: Implementation and Unit Testing 20](#_Toc27003372)

[Table 7 – Phase 4: Integration and System Testing 21](#_Toc27003373)

[Table 8 – Phase 5: Operation and Maintenance 21](#_Toc27003374)

[Table 9 - Use case LG.01 - <Guest> Login 28](#_Toc27003375)

[Table 10 - Use case BS.01 - <Babysitter> Get profile 31](#_Toc27003376)

[Table 11 - Use case BS.02 - <Babysitter> Edit sitting preferences 33](#_Toc27003377)

[Table 12 - Use case BS.03 - <Babysitter> Answer invitation 35](#_Toc27003378)

[Table 13 - Use case BS.04 - <Babysitter> Get invitation 37](#_Toc27003379)

[Table 14 - Use case BS.05 - <Babysitter> Get invitation list 39](#_Toc27003380)

[Table 15 - Use case BS.06 - <Babysitter> Get sitting history 41](#_Toc27003381)

[Table 16 - Use case PR.01 - <Parent> Create babysitting request 46](#_Toc27003382)

[Table 17 - Use case PR.02 - <Parent> Get babysitting request list 49](#_Toc27003383)

[Table 18 - Use case PR.03 - <Parent> Get babysitting request 51](#_Toc27003384)

[Table 19 - Use case PR.04 - <Parent> Accept babysitter 54](#_Toc27003385)

[Table 20 - Use case PR.05 - <Parent> Add people to trust circle 56](#_Toc27003386)

[Table 21 - Use case PR.06 - <Parent> Get request history 58](#_Toc27003387)

[Table 22 - Use case PR.07 - <Parent> Get profile 60](#_Toc27003388)

[Table 23 - Use case PM.01 - <Payment> Pay 62](#_Toc27003389)

[Table 24 - Use case PM.02 - <Payment> Refund 64](#_Toc27003390)

[Table 25 - Use case SH.01 - <System Handler > Generate identification tokens 67](#_Toc27003391)

[Table 26 - Use case SH.02 - <System Handler> Save transaction record 69](#_Toc27003392)

[Table 27 - Use case SH.03 - <System Handler> Matching parent and babysitter 73](#_Toc27003393)

[Table 28 - Use case SH.04 - <System Handler> Recommend babysitter to parent 75](#_Toc27003394)

[Table 29 - Use case ST.01 - <Staff> Create user account 78](#_Toc27003395)

[Table 30 - Use case ST.02 - <Staff> Update user information 81](#_Toc27003396)

[Table 31 - Use case ST.03 - <Staff> Ban user 83](#_Toc27003397)

[Table 32 - Use case ST.04 - <Staff> Edit configuration 85](#_Toc27003398)

[Table 33 - Front-end component diagram dictionary 89](#_Toc27003399)

[Table 34 - Back-end component diagram dictionary 90](#_Toc27003400)

[Table 35 – Class Dictionary 91](#_Toc27003401)

[Table 36 - Data Dictionary 100](#_Toc27003402)

[Table 37 – Data dictionary 102](#_Toc27003403)

|  |  |
| --- | --- |
| **Name** | **Definition** |
| **BSD** | Babysitting in Demand |
| **SPA** | Single Page Application |
| **2FA** | Two-Factor authentication |
| **DSDM** | Dynamic systems development method |

*Table 1: Definitions, Acronyms, and Abbreviations*

# **Outline**

## Project Information

Project name: **Babysitter in demand**

Project Code: **BSD**

Product Type: **Mobile Application**

Start Date: **September 10,2019.**

End Date: **--/--/2019.**

## Introduction

Taking care of children can be stressful. In modern days, parents are busy with their work and not many can spend time to properly look after their children or have the skills to take good care of their children.

Therefore, we believe finding a reliable babysitter is one of the problems that parents have to deal with nowadays.

We introduce a system capable of helping parents find suitable babysitter helping them taking care of their children. We verify the sitters using multi-factor authentication to ensure their identity and by connecting many parents together, parents can share their comment about the sitter they knew or hired.

## Current Situation

Parents who want to find babysitter usually use traditional methods. They rely on their relatives or friends to babysitting, they ask their relatives or friends to recommend them a babysitter. If neither their relatives and friends live near them nor they any have information, they will search for babysitter on the internet, websites, forums, social media... or go to the agency. After parents found babysitter, they will interview the babysitter then if they like the babysitter, parents and babysitter sign a contract together. Babysitter will be hired to work for a long period of time (3 months, 6 months...) they live in the same house with the family.

## Problem Definition

The current methods have the following advantages and disadvantages:

* **Advantages:**
* Relatives or friends are easy to contact and trustworthy.
* **Disadvantages:**
* **Relatives or friends are not always available** (Live far from parents, have work to do, being sick...).
* **Parents spend a lot of time to find and contact to the babysitter.**As mentioned in the introduction, finding babysitter can take a lot of time, parents have to go through various sources of information to find their ideal babysitter.
* **Babysitter recommended usually have little or no information.**Some basic information like: names, addresses, identification card, criminal record...
* **Parents don’t know what skills or experience babysitters have until they came.**Parents don’t know how experienced the babysitter are, their work history...
* **Parents can’t hire babysitter for a short time**Babysitter usually sign a long period contract for 3 – 6 months or more. Parents can’t hire them for just 1 or 2 hours if they want to.

## Proposed Solution

We proposed a mobile application to support parent to find a babysitter, based on their references and their circles of trust. The parent can choose which sitter they want when booking for a babysitting job. The system will suggest babysitters base on parent’s preferences, trust circles, and location. Babysitter have to go through 2 step verification before taking their babysitting job. By conducting surveys and interviews on District 1, 2, Phu Nhuan, we see that the younger generation have positive feedback about the application.

### Feature functions

* **Staff side:**
* **Approve Babysitter profile:** Staff check information and provide Babysitter with an e-token.
* **Approve Parent profile:** Staff check information.
* **Parent side:**
* **Request sitter:** Parent can create and cancel babysitting request.
* **Accept babysitter offer:** Choose and accept the offer from babysitters that accept your request.
* **Create trust circl**e: Parents can find babysitter through a trust circle created by parent.
* **Sitter side:**
* **Verify before accept a request**: Login by e-token provided by Admin.
* **Accept the invitation:**Get invitations by parent.
* **System handler:**
* **Suggest babysitter**: Automatically suggest babysitter that match parent's criteria.

### Values and challenges

* **Values:**
* **Parent Users:**
* Provide service to support busy people take care of their kids but still can do their jobs in the same times.
* Help parents get some free time to take care of themselves.
* Decrease stresses, overloaded in family matters, save a lot of times for busy people, make user feels more free, comfortable to do somethings else like hang out, shopping, relax, for relationship, etc.
  + **Babysitter Users:**
* Provide an opportunity job for who looking for babysitter job.
* Easier and faster supply babysitter finders in the whole city.
* **Challenges:**
* **System:**
* Babysitter account maybe use by who is not registered.
* Safety for both Parent users and babysitter users.
* Rating collect by users maybe not exactly represent the ability of the babysitter (fake data, wrong data...).
* **Parent Users:**
* Trust of users when hires the babysitter.
* Hard to booking in holidays because of the increment of needs.
* **Babysitter Users:**
* Supplies jobs as many as users need in a regular way.

## Functional Requirements

Functional requirements of the system are listed as below:

### Staff Component

* Create user account
* Manage user
* Manage configurations

### Parent Component

* Create a babysitting request
* Find babysitters based on criteria
* Cancel a babysitting request
* View the sitting history
* View profile
* Pay for the sitting
* Create a trusted circle that include other parents they trust

### Babysitter component

* Login using provided e-token
* View their personal information
* View and edit their sitting preferences
* Answer to an invitation from parent using e-token

### Authentication component

* Provide babysitter with an E-token when they have been accepted to the system
* Require the babysitter to verify themselves with the E-token provided.

### Recommendation component

* Recommend suitable babysitters based on the parent’s criteria and the babysitter sitting preferences
* Recommend suitable babysitter for the parent based on their trusted circle

## Roles and Responsibilities

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No** | **Full Name** | **Role** | **Position** | **Contact** |
| 1 | Kiều Trọng Khánh | Project Manager | Supervisor | khanhkt@fpt.edu.vn |
| 2 | Phùng Thiên Phúc | Developer | Leader | phucptse62639@fpt.edu.vn |
| 3 | Phạm Hải Dương | Developer | Member | duongphse62746@fpt.edu.vn |
| 4 | Hồ Tấn Kỳ | Developer | Member | kyhtse61974@fpt.edu.vn |
| 5 | Hoàng Nhật Đông | Developer | Member | donghnse62357@fpt.edu.vn |

Table 1 - Roles and Responsibilities

# **Software Project Management Plan**

## Problem Definition

### Name of this Capstone Project

**Official name**: Babysitter in demand

**Vietnamese name**: Ứng dụng hỗ trợ tìm người trông trẻ theo nhu cầu

**Abbreviation**: BSD

### Problem Abstract

Currently in Viet Nam, especially in big city like Ho Chi Minh, parents are busy with their work, they have the needs to hire a babysitter that they can trust but they don’t have much time to carefully looking for a babysitter.

We provide a software to support them to find a trustworthy babysitter, the system aims to provide parents and babysitters a better place in which they can find each other through the connection between others parents. The system also supports parents to hire a babysitter suited with their needs. However, our system must resolve or at least reduce the trust issues between parents and babysitters. For example, parent can verify babysitter with face recognition, voice recognition or fingerprint when they show up for the job, babysitter have to go through 2FA (two-factor authentication) process.

We need to build this system as secure as possible but also keep it simple enough so that the user can easily interact with it (babysitter may not have great knowledge about technology).

### Project Overview

#### Current Situation

Below are some problems that is existing or might encounter in the process:

* **Lack of business view**: all team members are specialized in software engineering, leading to a lack of knowledge in the current library management process. Therefore, it will take extra time to learn and offer solutions to the current problems of the library management system.
* **The majority of the team isn’t familiar with technologies use in this project**: JavaScript frameworks (ReactJs, React Native, Nodejs). The team need time to research and get acquainted with these technologies.
* **Security**: How can our system ensured user’s data? How to verify the babysitter identity on the application?
* **Availability**: The server can be down or attacked.
* Team member leave or show lack of productivity.
* Consider the trade-off between secure and utility

#### The Proposed System

We create a mobile application in client side for Parent users and Babysitter users, 1 web application for our staffs to manage accounts, tracking all happen transactions, support and receive feedbacks from clients. That will help us to provide services to our customers and improve user experiences from collected feedbacks.

We build a system bases on JavaScript frameworks, using one language to build both front and backend, this help to reduce the cost of dealing with different technology.

For the user experience we use React Native, with React Native, our application will display consistent on both IOS and Android. Being a component-based, it helps us to write code that can be re-use and thus reduce the time to debug and leave developer with more time to focus on the core features.

#### Boundaries of the System

**What the system does:**

* + For the staffs**:**
* Manage the parent/babysitter account:
* Verify and create the babysitter/parent account.
* Edit babysitter account information.
* Lock/Unlock account.
* View babysitting history from account
* Receive feedbacks from users.
  + For the Babysitter:
* View list of available babysitting requests and choose which one to babysit
* View incoming babysittings
* View account’s information
* View and edit sitting preferences
* Accept babysitting invitation from parent using e-token
  + For the Parent**s:**
* Create babysitting invitation to find the babysitter with some criteria
* View account’s information
* Send feedback about the babysitter
* View the sittings history
* Create circles of trust
* Pay for babysitting via the application
  + - For the System Handler
    - Recommend babysitter when parent request babysitting
    - Calculate babysitter point base on their preferences and rating
* What the system do not:
  + System do not guarantee the accuracy, reliability, suitability, timeliness, completeness, performance or fitness for a particular purpose of the Application or any material and information available through the Application
  + System do not handle babysitter

#### Future plans

* System will allow parents who have the same criteria, preferences to request one babysitting
* System support pay by cash
* Sitter can send babysitting request to the parent’s sitting request

## Project organization

### Software Process Model

As mentioned above, the team have to deal with the following situations:

* Working on the very sensitive problem, children, we want to make sure we can support the parent to find a suitable babysitter. Because so, we need to get user involve to the development of the application as soon as possible, parent and babysitter will help testing the features of the application and give us feedback to improve our system.
* Two of the team members are not familiar with the JavaScript framework which we use in this project. We need short review and testing to make sure the quality of the code.
* For the above reasons, the team decided to follow the DSDM model which will help us:
* Deal with possible changes in requirements.
* Automation the deployment and testing
* Continuously work on the cycle of researching – applying – testing – deploy when facing new technology (JavaScript frameworks for both front-end and back-end).
* Focus on Developing new features rather than fix bug, design and documentation.

The project-lifecycle can be divided into four phases:

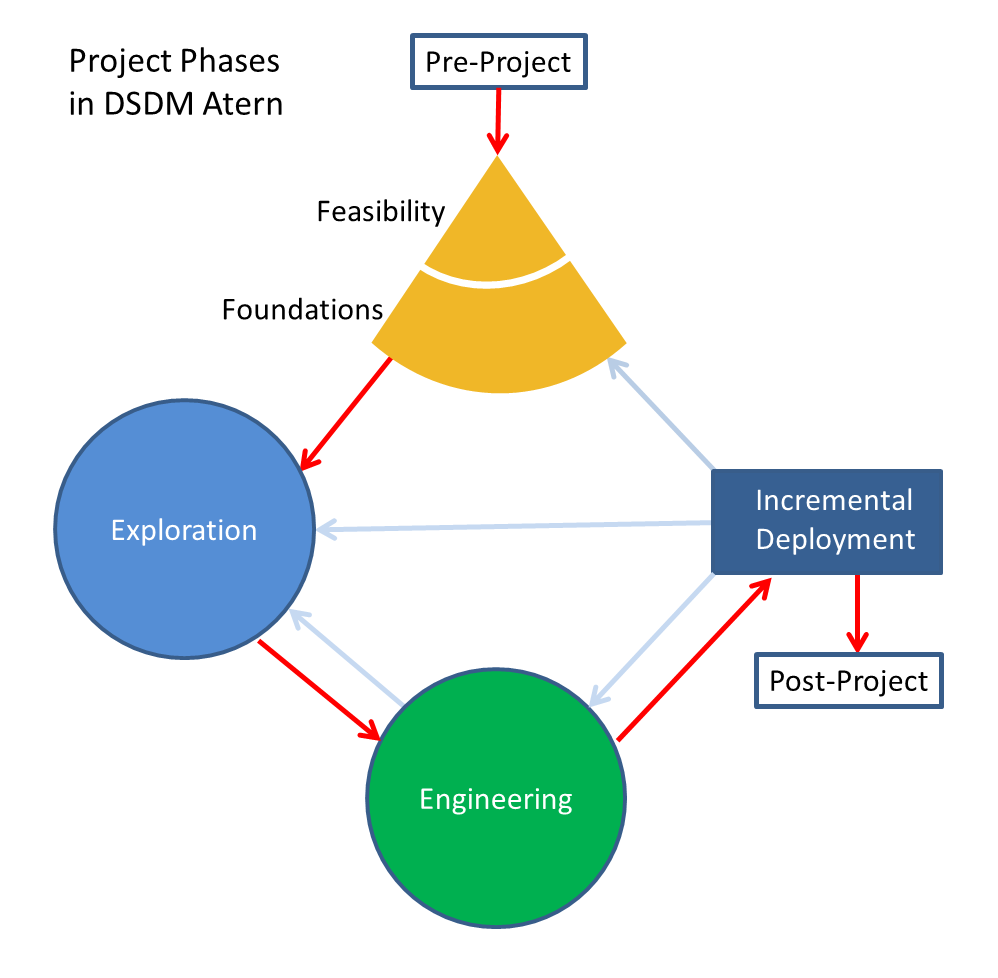


Figure 1 - Atern Development Model – DSDM (1)

|  |  |
| --- | --- |
| Phase | Key Responsibilities |
| Pre-project | Initiation of the project, agreeing the Terms of Reference for the work |
| Feasibility | the viability and the outline business case (justification). |
| Foundations | Key phase for ensuring the project is understood and defined well enough so that the scope can be baselined at a high level and the technology components and standards agreed, before the development activity begins. |
| Exploration | Iterative development phase during which teams expand on the high-level requirements to demonstrate the functionality |
| Engineering | Iterative development phase where the solution is engineered to be deployable for release |
| Deployment | For each Increment (set of timeboxes) of the project the solution is made available. |
| Post project | Assesses the accrued benefits. |

Table 2 - Key responsibilities

## Project Management Plan

### Software development life cycle

Figure 2 – Software development life cycle[1]

Below are all the major tasks that need to be performed sequentially during the development of the system.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Phase** | **Description** | **Deliverables** | **Resource needed** | **Dependencies and Constrains** | **Risk** |
| **Requirements Definition** | Identify and clarify the system requirements | Report No.1 Introduction | 20 days | N/A | Missing requirement  Project’s scope can be unclear  Lack of member share and understand |
| **System and Software Design** | Identify hardware and software requirements  Decide software architect and clarify software detail design  Design database | Report No.2 Software Project Management Plan  Report No. 3 Software Requirement Specification  Report No. 4 Software Design Description | 50 days | Depend on Requirements Definition | Misunderstood or unclear system’s requirement  Lack of practical experience leading to unreasonable design |
| **Implementation and Unit Testing** | Implements all functions of system  Create test plan  Perform Unit testing | Software package | 120 days | Base on Software Requirement Specification and Software Design Description  Coding try to follow coding convention | Member does not perform unit test  Lack of practical experience |
| **Integration and System Testing** | Perform integration test and system test | Report No. 5 System Implementation & Test | 35 days | Implementation and Unit Testing are finished | Lack of testing experience leading to lack of test cases  Not enough time for performing test |
| **Operation and Maintenance** | Deploy the system  Create the user’s manuals  Do routine maintenance activities | Report No.6 Software User’s Manual | 15 days | Integration and System Testing are finished | User’s manual may be difficult for user to understand and confuse |

Table 3 – Phase overview

### Phase Detail

#### Phase 1: Requirements Definition

|  |  |  |
| --- | --- | --- |
| Task | Description | Author |
| Identify and clarify system requirements. | Research current systems to collect requirements  Define main and needed functions the system must include | Phùng Thiên Phúc  Phạm Hải Dương  Hoàng Nhật Đông  Hồ Tấn Kỳ |

Table 4 - Phase 1: Requirement Definition

#### Phase 2: System and Software Design

|  |  |  |
| --- | --- | --- |
| Task | Description | Author |
| Identify hardware and software requirements | Find out the suitable hardware and software for the system, as well as its minimum and recommended requirements | Phùng Thiên Phúc  Phạm Hải Dương  Hoàng Nhật Đông  Hồ Tấn Kỳ |
| Decide software architect and clarify software detail design | Define the major software components and interfaces.  Draw core-flow diagram, use case diagram, prototype …  Group meeting to review and modify | Phùng Thiên Phúc  Phạm Hải Dương  Hoàng Nhật Đông  Hồ Tấn Kỳ |
| Design database | Design database for the system | Phùng Thiên Phúc  Phạm Hải Dương |

Table 5 – Phase 2: System and Software Design

#### Phase 3: Implementation and Unit Testing

|  |  |  |
| --- | --- | --- |
| Task | Description | Author |
| Implements all functions of system | Coding all the components | Phùng Thiên Phúc  Phạm Hải Dương  Hoàng Nhật Đông  Hồ Tấn Kỳ |
| Create test plan | Planning for testing | Phùng Thiên Phúc  Phạm Hải Dương  Hoàng Nhật Đông  Hồ Tấn Kỳ |
| Perform Unit testing | Write Unit test cases  Implement Unit tests | Phùng Thiên Phúc  Phạm Hải Dương  Hoàng Nhật Đông  Hồ Tấn Kỳ |

Table 6 – Phase 3: Implementation and Unit Testing

#### Phase 4: Integration and System Testing

|  |  |  |
| --- | --- | --- |
| Task | Description | Author |
| Perform integration test and system test | Test groups of modules and test whole the system | Phùng Thiên Phúc  Phạm Hải Dương  Hoàng Nhật Đông  Hồ Tấn Kỳ |

Table 7 – Phase 4: Integration and System Testing

#### Phase 5: Operation and Maintenance

|  |  |  |
| --- | --- | --- |
| Task | Description | Author |
| Deploy the system | Deploy the system in client environment | Phùng Thiên Phúc  Phạm Hải Dương  Hoàng Nhật Đông  Hồ Tấn Kỳ |
| Create the user’s manuals. | Create a guideline to instruct users using system | Phùng Thiên Phúc  Phạm Hải Dương  Hoàng Nhật Đông  Hồ Tấn Kỳ |
| Do routine maintenance activities. | Do routine maintenance activities for client system | Phùng Thiên Phúc  Phạm Hải Dương  Hoàng Nhật Đông  Hồ Tấn Kỳ |

Table 8 – Phase 5: Operation and Maintenance

### Task sheet

Refer to “Task sheet” folder: [https://bom.to/RQAoID](https://bom.to/RQAoID?fbclid=IwAR1DWxppa1nLHg5tucgR7VLYkAGciePjPPHI4gl4TQDs1kIrVMYlv3SzGUI)

### All Meeting Minutes

Refer to “Meeting minutes” folder: [https://bom.to/RQAoID](https://bom.to/RQAoID?fbclid=IwAR1DWxppa1nLHg5tucgR7VLYkAGciePjPPHI4gl4TQDs1kIrVMYlv3SzGUI)

## Coding Convention

Javascript ES6: Using to develop the project.

**Summary:**

**Naming Convention:**

Variable names should be short yet meaningful. The choice of a variable name should be designed to indicate to the casual observer the intent of its use.

Methods should be verbs, in mixed case with the first letter lowercase, with the first letter of each internal word capitalized.

**Indentation:**

One declaration per line is recommended since it encourages commenting.

In absolutely no case should variables and functions be declared on the same line.

Do not put different types on the same line.

**Declarations Convention:**

One declaration per line is recommended since it encourages commenting.

Using Airbnb JavaScript Style Guide from:

<https://github.com/airbnb/javascript>

# **Software Requirement Specification**

## User Requirement Specification

### Guest requirement

Guest is a person who doesn’t have access to the system. Guest can only use some functions in the system. To use all functions, guest must login. There are some functions guest can use.

* Login

### Parent requirement

Parent is a person who uses the application’s service. The parent can use some following functions:

* Create the babysitting request
* Answer the babysitting request
* Get the profile information
* Manage trust circle
* Create the trust circle
* Join the trust circle
* Pay for the sitting
* Get sitting history
* Send feedback for the sitting

### Babysitter requirement

Babysitter is a person who uses the application’s service. The parent can use some following functions:

* Answer parent’s invitation
* Get incoming babysitting
* Get sitting history
* Get invitation list
* Edit preferences

### Staff requirement

Staff is a person who works with the system to manage:

* Manage users
* Create the user
* Edit/Update profile of the user
* Deactivate the user (ban)
* Change configuration

### System handler requirement

* Recommend babysitter
* Provide e-token to babysitter
* Match parent and babysitter

### Authorized User

* Logout

### Payment

* Pay the transaction

## System Requirement Specification

### System Overview Use Case

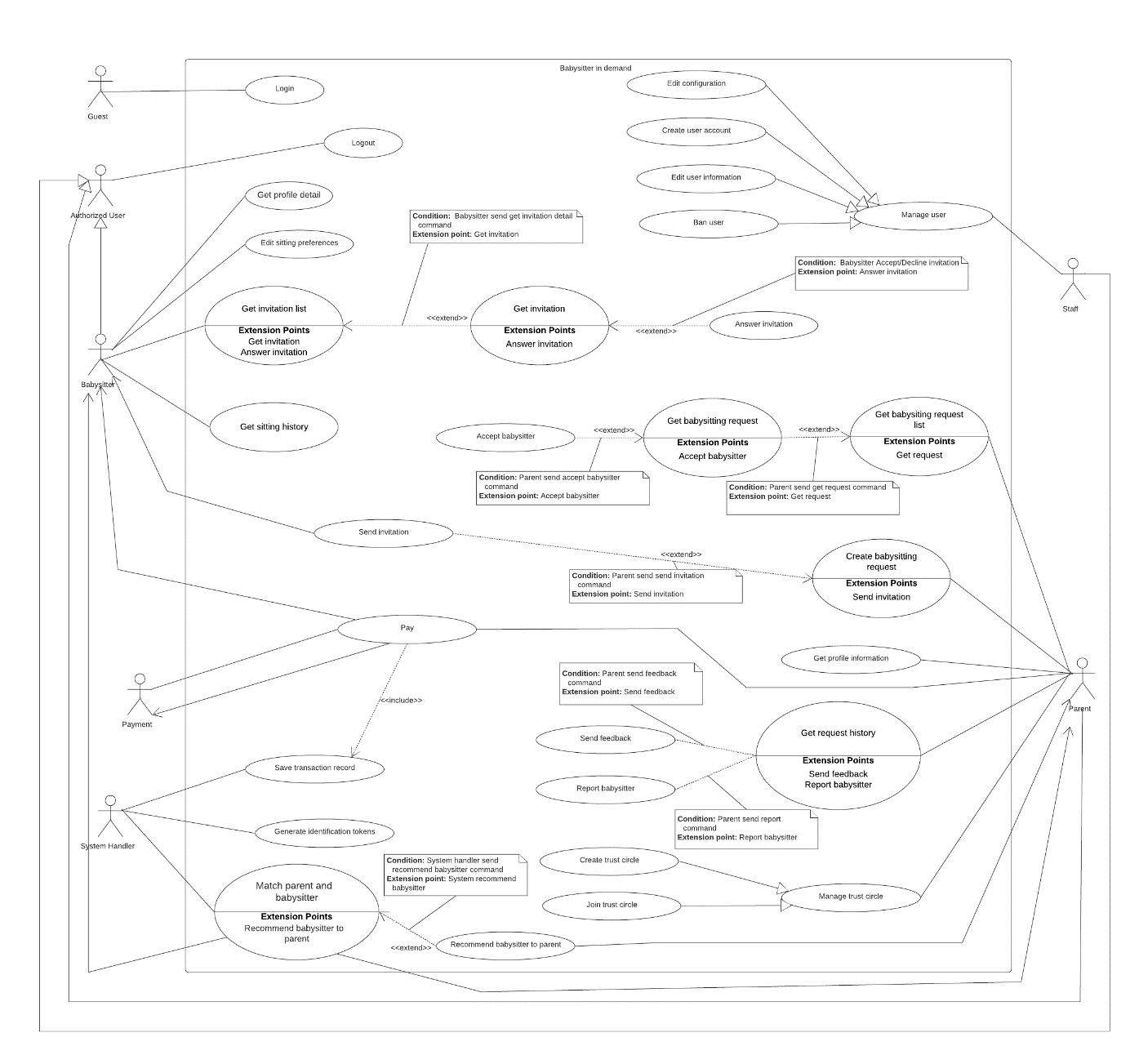


Figure 3 – Overview Use Case

### List of Use Case

##### <Guest> Login

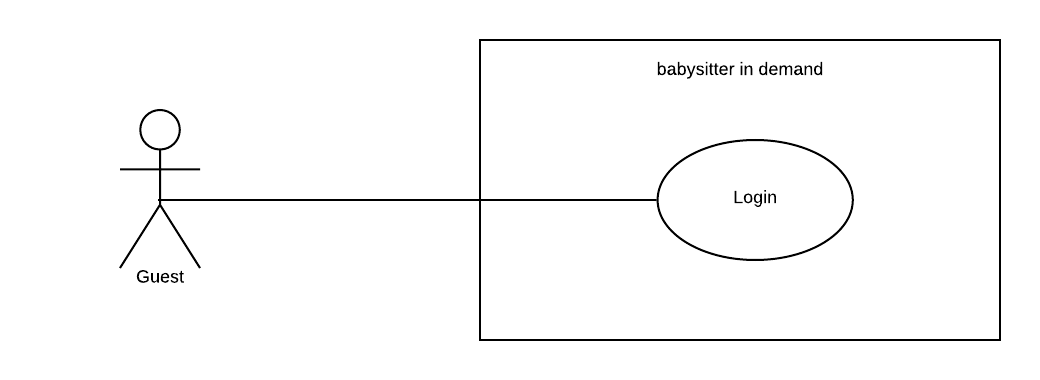


Figure 4 - <Guest> Login

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – UC\_LG.01** | | | |
| **Use Case No.** | UC\_LG.01 | **Use Case Version** | 2.0 |
| **Use Case Name** | Login | | |
| **Author** | PhucPT | | |
| **Date** | 27/09/2019 | **Priority** | Normal |
| **Actor:**   * Guest   **Summary:**   * This use case allows Guest user to be verified if they have the right to access the system or not * The actor starting this use case are parent, babysitter, staff and admin   **Goal:**   * Verified if Guest is a member of the system   **Triggers:**   * Guess send login command to the system   **Preconditions:**   * N/A   **Post Conditions:**   * **Success:**   + Guest have been verified * **Fail**:   + Show error message.   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | **Step** | **Actor Action** | **System Response** | | 1 | Guest provide identity information   * Username * Password   Guest send command to login into the system except babysitter role  **[Alternative 1] [Alternative 2]** | System verify the identity information that Guess provide  **[Exceptions 1]**  System allow Guest to access to the system |   **Alternative Scenario:**  **[Alternative 1]**   |  |  |  | | --- | --- | --- | | **Step** | **Actor Action** | **System Response** | | 1 | Guest provide identity information   * Username * Password * Authentication code   Guest send command to login into the system with the babysitter role | System verify the username and password that Guess provide  **[Exceptions 1]**  System also require the Authentication code  **[Exceptions 1]**  System allow Guest to access to the system |   **[Alternative 2]**   |  |  |  | | --- | --- | --- | | **Step** | **Actor Action** | **System Response** | | 1 | Guest does not have account in the system yet | System show error message that user not found |   **Exceptions:**   |  |  |  | | --- | --- | --- | | **No** | **Cause** | **System Response** | | 1 | Wrong or Expire Authentication code | System show error and require user to re-input |   **Relationships:** N/A  **Business Rules:**   * Guest is user that haven’t been verified by the system * Guest must have an account in the system in order to login into it * Staff is a user who work on our system * Admin is a user who manage system configuration * Parent is a user who want to find babysitter * Babysitter is a user who want to find babysitting job * Guest with role other than “babysitter” will use username and password to login into the system * Guest with “babysitter” role need to provide their username and password and also their authentication code to login into the system * Authentication code is the code that generated from their devices Authentication app (Twilio Authy or Google Authenticator) * If identification information is valid system allow user to access * If identification information is invalid system show error message | | | |

Table 9 - Use case LG.01 - <Guest> Login

##### <Babysitter> Overview Use Case

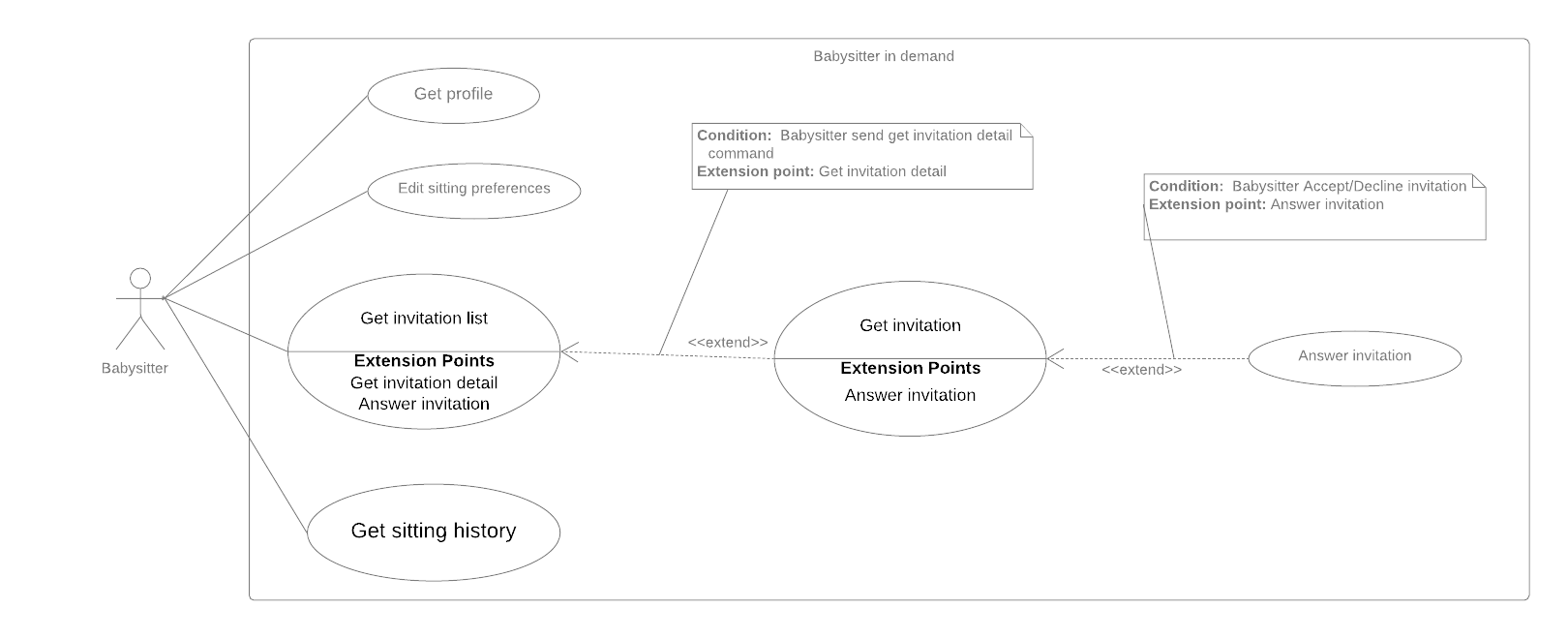


Figure 5 - <Babysitter> Overview Use Case

###### <Babysitter> Get profile

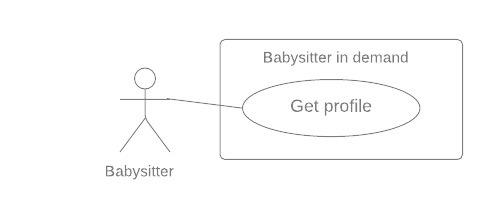


Figure 6 - <Babysitter> Get profile

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – UC\_BS.01** | | | |
| **Use Case No.** | UC\_BS.01 | **Use Case Version** | 2.0 |
| **Use Case Name** | Get profile | | |
| **Author** | KyHT | | |
| **Date** | 27/09/2019 | **Priority** | Normal |
| **Actor:**   * Babysitter   **Summary:**   * This use case allows babysitter to get profile.   **Goal:**   * System returns babysitter profile.   **Triggers:**   * Babysitter sends command to get profile.   **Preconditions:**   * User must login to the system with the role “Babysitter”. * Information has been checked by staff.   **Post Conditions:**   * **Success:**    + System returns babysitter profile. * **Fail**:   + Show error message.   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | **Step** | **Actor Action** | **System Response** | | 1 | Babysitter sends command to get profile. | System returns babysitter profile detail by Id.  **[Exceptions 1]** | | | | |
| **Alternative Scenario:** N/A  **Exceptions:**   |  |  |  | | --- | --- | --- | | **No** | **Cause** | **System Response** | | 1 | Error when get babysitter profile | System show error message |   **Relationships:** N/A  **Business Rules:**   * Babysitter profile detail has the following:   + - Profile picture     - First name     - Last Name     - Birth Day     - Gender     - Address     - Work schedule * All information cannot be modified by babysitter. * Personal information is only changed when verified by BSD staff. | | | |

Table 10 - Use case BS.01 - <Babysitter> Get profile

###### <Babysitter> Edit sitting preferences

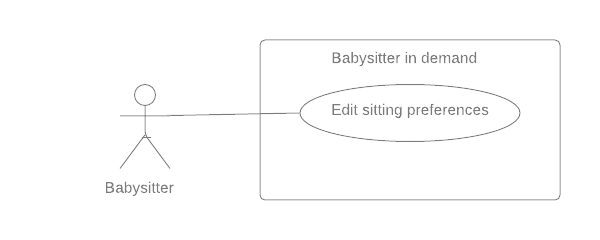


Figure 7 - <Babysitter> Edit sitting preferences

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **USE CASE – UC\_BS.02** | | | | | |
| **Use Case No.** | UC\_BS.02 | **Use Case Version** | | 2.0 |
| **Use Case Name** | Edit sitting preferences | | | | |
| **Author** | KyHT | | | | |
| **Date** | 27/09/2019 | **Priority** | Normal | | |
| **Actor:**   * Babysitter   **Summary:**   * This use case allows babysitter to edit sitting preferences.   **Goal:**   * System updates new babysitter sitting preferences.   **Triggers:**   * Babysitter sends a command to edit babysitter sitting preferences.   **Preconditions:**   * User must login to the system with the role “Babysitter”.   **Post Conditions:**   * **Success:**    + System updates new babysitter sitting preferences. * **Fail**:   + Show error message.   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | **Step** | **Actor Action** | **System Response** | | 1 | Babysitter sends command to edit babysitter preferences.  **[Alternative 1]** | The system displays previously saved sitting preferences. | | 2 | Babysitter sends command to save new babysitter preferences.  **[Exceptions 1]** | System notify the babysitter about the success  System update new preferences | | | | | | |
| **Alternative Scenario:**  **Alternative 1:**   |  |  |  | | --- | --- | --- | | **Step** | **Actor Action** | **System Response** | | 1 | Babysitter sends command to cancel edit babysitter preferences. | System returns to the Setting screen. |   **Exceptions:**   |  |  |  | | --- | --- | --- | | **No** | **Cause** | **System Response** | | 1 | Error when saving babysitter preferences. | System show error message |   **Relationships:** N/A  **Business Rules:**   * Sitting preferences detail include:   + Work schedule   + Start time – End time * All this information can be edited later. | | | | | |

Table 11 - Use case BS.02 - <Babysitter> Edit sitting preferences

###### <Babysitter> Answer invitation

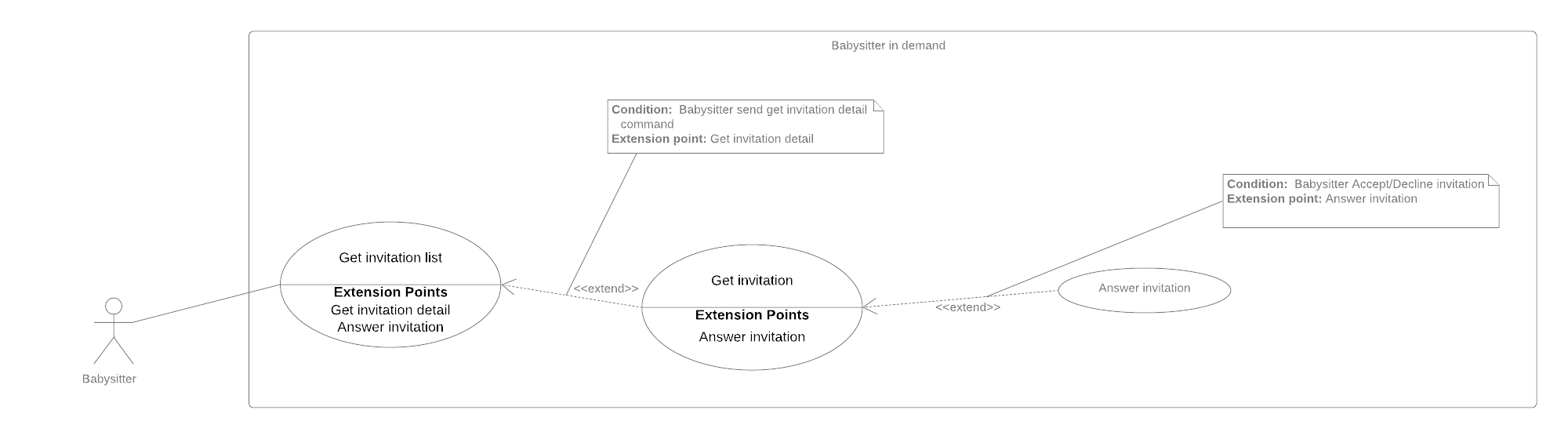


Figure 8 - <Babysitter> Answer invitation

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **USE CASE – UC\_BS.03** | | | | |
| **Use Case No.** | UC\_BS.03 | **Use Case Version** | | 2.0 |
| **Use Case Name** | Answer invitation | | | |
| **Author** | KyHT | | | |
| **Date** | 27/09/2019 | **Priority** | High | |
| **Actor:**   * Babysitter   **Summary:**   * This use case allows a babysitter to answer invitation.   **Goal:**   * Babysitter can “Accept” or “Decline” invitation * System notify to babysitter whether they are accepted or decline   **Triggers:**   * Babysitter sends command to answer invitations.   **Preconditions:**   * User must login to the system with the role of “Babysitter”. * The parent has sent an invitation to the babysitter. * The sitting request haven’t been confirmed and must be in “Pending” state.   **Post Conditions:**   * **Success:**   + Babysitter can “Accept” or “Decline” invitation   + System notify to babysitter whether they are accepted or decline * **Fail**:   + Show error message   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | **Step** | **Actor Action** | **System Response** | | 1 | Babysitter send accept invitation command  **[Alternative 1]** | System change the parent’s babysitting request state to “Accepted”  System notify the parent about the success |   **Alternative Scenario:**  **Alternative 1**   |  |  |  | | --- | --- | --- | | **Step** | **Actor Action** | **System Response** | | 1 | Parent send decline invitation command | System change the parent’s babysitting request state to “Decline”  System notify the parent about the change |   **Exceptions:** N/A  **Relationships:**   * **Get invitation** extends **Get invitation list**. * **Answer** extends **Get** **invitation list.**   **Business Rules:**   * The parent’s request must be at “Pending” state, indicate the request is waiting for a babysitter * When the babysitter accepts an invitation from a parent, the parent babysitting request state will be change to “Accepted” and the babysitter invitation will also change state to “Accepted”. * When the babysitter declines an invitation from a parent, the parent babysitting request state will be change to “Decline” and the babysitter invitation will also change state to “Decline”. * Information about the state change will be notify to the parent. | | | | |

Table 12 - Use case BS.03 - <Babysitter> Answer invitation

###### <Babysitter> Get invitation

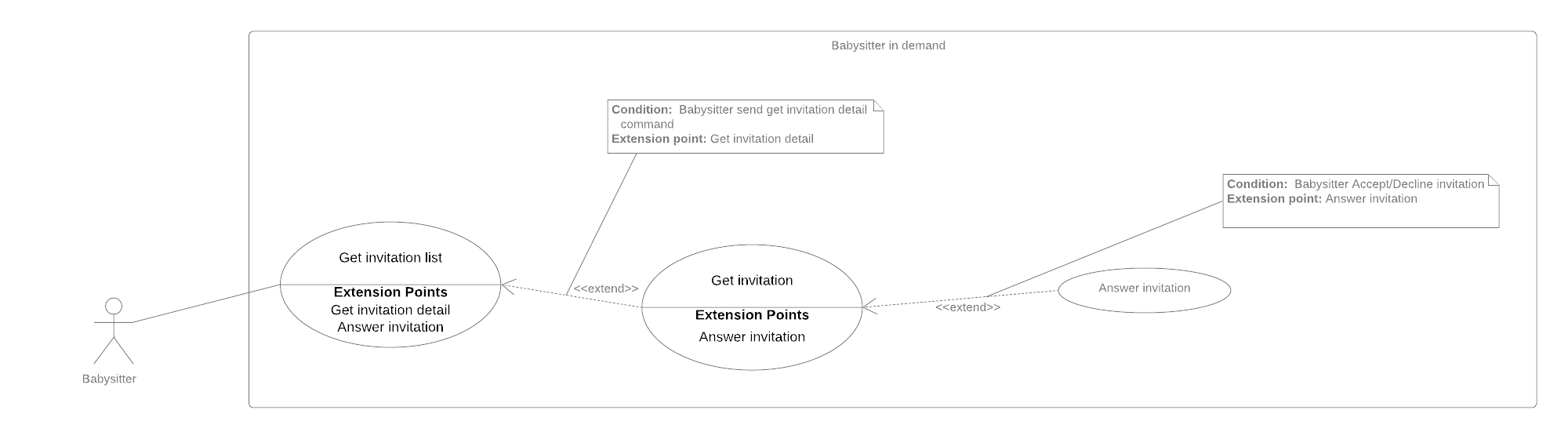


Figure 9 - <Babysitter>Get invitation

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **USE CASE – UC\_BS.04** | | | | |
| **Use Case No.** | UC\_BS.04 | **Use Case Version** | | 2.0 |
| **Use Case Name** | Get invitation | | | |
| **Author** | KyHT | | | |
| **Date** | 27/09/2019 | **Priority** | High | |
| **Actor:**   * Babysitter   **Summary:**   * This use case allows babysitter to get invitation.   **Goal:**   * System returns invitation.   **Triggers:**   * Babysitter sends command to get invitation.   **Preconditions:**   * User must login to the system with the role “Babysitter”.   **Post Conditions:**   * **Success:**    + System returns detail invitation. * **Fail**:   + Show error message   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | **Step** | **Actor Action** | **System Response** | | 1 | Babysitter sends command to get invitation.  **[Exceptions 1]** | System returns detail invitation. |   **Alternative Scenario:** N/A  **Exceptions:**   |  |  |  | | --- | --- | --- | | **No** | **Cause** | **System Response** | | 1 | Error when send command to get invitation detail. | System show error message. |   **Relationships:**   * **Get invitation** extends **Get invitation list**. * **Answer** extends **Get** **invitation list.**   **Business Rules:**   * Details invitation of parent has the following:   + Profile parent   + Date   + Start time   + End time   + Sitting address   + Number of children   + Youngest age   + Payment   + Status * Babysitter can accept the invitation or decline the invitation when invitation has status “Pending”. | | | | |

Table 13 - Use case BS.04 - <Babysitter> Get invitation

###### <Babysitter> Get invitation list

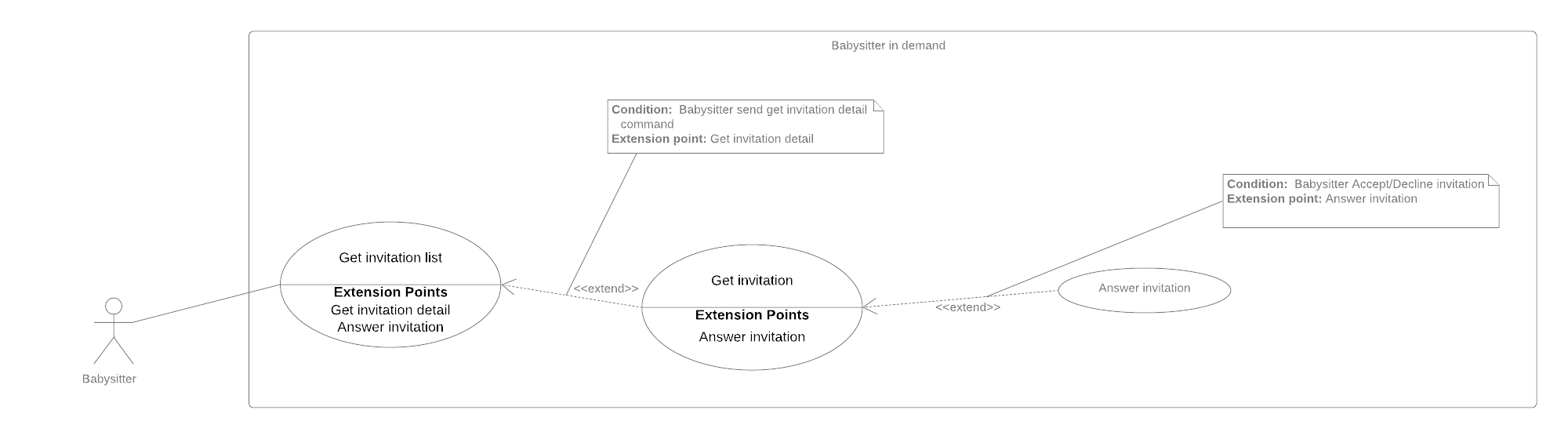


Figure 10 - <Babysitter> Get invitation list

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **USE CASE – UC\_BS.05** | | | | |
| **Use Case No.** | UC\_BS.05 | **Use Case Version** | | 2.0 |
| **Use Case Name** | Get invitation list | | | |
| **Author** | KyHT | | | |
| **Date** | 27/09/2019 | **Priority** | High | |
| **Actor:**   * Babysitter   **Summary:**   * This use case allows a babysitter to get invitation list.   **Goal:**   * System returns list of invitations.   **Triggers:**   * Babysitter sends command to get list invitations.   **Preconditions:**   * User must login to the system with the role “Babysitter”.   **Post Conditions:**   * **Success:**    + System returns list invitations. * **Fail**:   + Show error message   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | **Step** | **Actor Action** | **System Response** | | 1 | Babysitter sends command to get a list of invitation. | System check invitations to babysitter  **[Alternative 1] [Exceptions 1]**  System returns a list of invitation which sent to the babysitter. |   **Alternative Scenario:**  **Alternative 1:**   |  |  |  | | --- | --- | --- | | **Step** | **Actor Action** | **System Response** | | 1 | No invitation to show | System notify the babysitter that there is no invitation. |   **Exceptions:**   |  |  |  | | --- | --- | --- | | **No** | **Cause** | **System Response** | | 1 | Error when send command to get invitation detail. | System show error message. |   **Relationships:**   * **Get invitation** extends **Get invitation list**. * **Answer** extends **Get** **invitation list.**   **Business Rules:**   * The system will display list babysitting in order of the latest date. * Babysitter invitations contain the following information: * Name parent * Date   + Price   + Distance   + Status (“Pending” and “Confirm”) | | | | |

Table 14 - Use case BS.05 - <Babysitter> Get invitation list

###### <Babysitter> Get sitting history

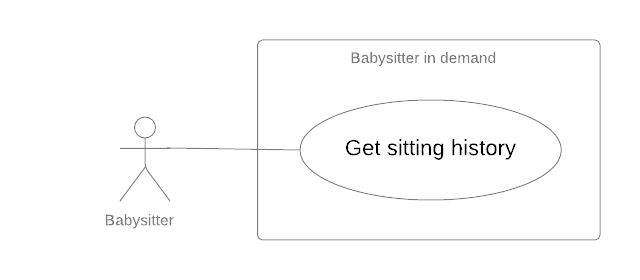


Figure 11 - <Babysitter> Get sitting history

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **USE CASE – UC\_BS.06** | | | | |
| **Use Case No.** | UC\_BS.06 | **Use Case Version** | | 2.0 |
| **Use Case Name** | Get sitting history | | | |
| **Author** | PhucPT | | | |
| **Date** | 27/09/2019 | **Priority** | Normal | |
| **Actor:**   * Babysitter   **Summary:**   * This use case allows a babysitter to get sitting history.   **Goal:**   * System returns history of sitting.   **Triggers:**   * Babysitter sends command to get history of sitting.   **Preconditions:**   * User must login to the system with the role “Babysitter”.   **Post Conditions:**   * **Success:**    + System returns history of sitting. * **Fail**:   + Show error message   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | **Step** | **Actor Action** | **System Response** | | 1 | Babysitter sends command to get history of sitting. | System check the history of sitting.  **[Alternative 1]**  System returns history of sitting. | | | | | |
| **Alternative Scenario:**  **Alternative 1:**   |  |  |  | | --- | --- | --- | | **Step** | **Actor Action** | **System Response** | | 1 | No history to show | System show message the “No sitting history yet” when the babysitter has not yet accepted/completed any offers. |   **Exceptions:**   |  |  |  | | --- | --- | --- | | **No** | **Cause** | **System Response** | | 1 | Error when get history of sitting | System show error message |   **Relationships:** N/A  **Business Rules:**   * System displays list history in the order of the latest date. * The history list stores requests with status: * DONE * PARENT\_CANCELED * DONE\_UNCONFIRMED * DONE\_BY\_NEW\_START * SITTER NOT CHECKIN * EXPIRED | | | | |

Table 15 - Use case BS.06 - <Babysitter> Get sitting history

##### <Parent> Overview Use Case

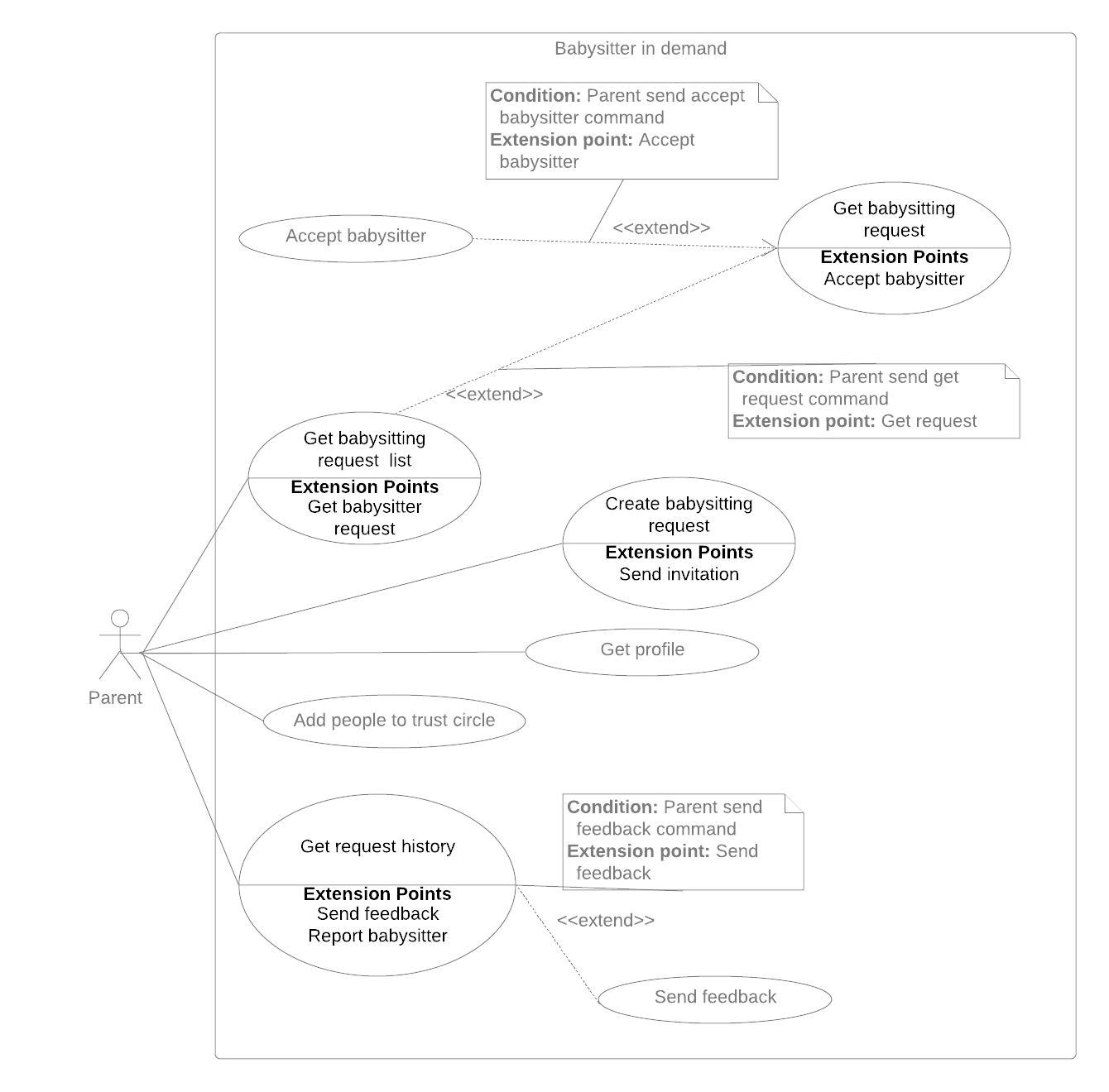


Figure 12 - <Parent> Overview Use Case

###### <Parent> Create babysitting request

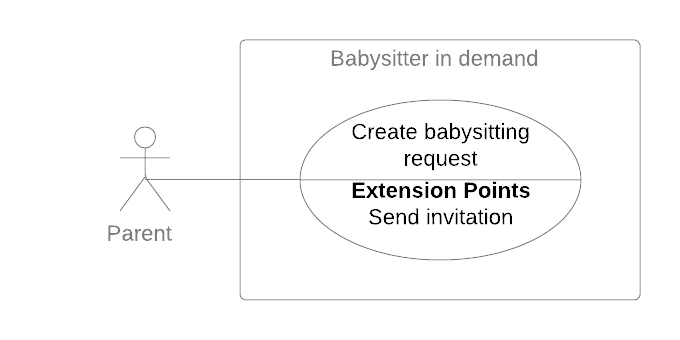


Figure 13 - <Parent> Create babysitting request

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – UC\_PR.01** | | | |
| **Use Case No.** | UC\_PR.01 | **Use Case Version** | 2.0 |
| **Use Case Name** | Create babysitting request | | |
| **Author** | DuongPH | | |
| **Date** | 26/11/2019 | **Priority** | High |
| **Actor:**   * Parent   **Summary:**   * This use case allows parent to create new babysitting request.   **Goal:**   * To create a new babysitting request   **Triggers:**   * Parent sends command to create new babysitting request.   **Preconditions:**   * User must log into the system with account has role Parent.   **Post Conditions:**   * **Success:**    + System create a babysitting request.   + System sends invitation to the babysitter. * **Fail**:   + Show error message.   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | **Step** | **Actor Action** | **System Response** | | 1 | Parent sends command to create new babysitting request. | System renders new request form. | | 2 | Parent inputs into information fields.  **[Alternative 1] [Alternative 2]** |  | | 3 | Parent sends command to get recommendation for the request. | System return a list of matched babysitters.  **[Alternative 3]** | | 4 | Parent sends command to invite a babysitter. | System check if parent has provided a credit card.  **[Alternative 4]**  If the parent has provided a credit card before, system will create a babysitting request with status “Pending”.  System send an invitation to the babysitter.  **[Exception 1, 2]** |   **Alternative Scenario:**  **Alternative 1:**   |  |  |  | | --- | --- | --- | | **Step** | **Actor Action** | **System Response** | | 1 | Parent didn’t provide the sitting time | System shows message to notify the parent. |   **Alternative 2:**   |  |  |  | | --- | --- | --- | | **Step** | **Actor Action** | **System Response** | | 1 | Parent didn’t choose any child | System shows message to notify the parent. |   **Alternative 3:**   |  |  |  | | --- | --- | --- | | **Step** | **Actor action** | **System Response** | | 1 | The sitting request detail is not matched with any babysitter’s preferences | System shows message to notify the parent. |   **Alternative 4:**   |  |  |  | | --- | --- | --- | | **Step** | **Actor Action** | **System Response** | | 1 | Parent didn’t provide a credit card | System ask the parent to provide a credit card. | | 2 | Parent enter credit card information | System check the card information.  System will create a babysitting request with status “Pending”.  System send an invitation to the babysitter.  **[Exception 1, 2]** |   **Exceptions:**   |  |  |  | | --- | --- | --- | | **No.** | **Cause** | **System Response** | | 1 | Failed to create request. | System shows error message. | | 2 | Failed to send invitation to babysitter. | System shows error message. |   **Relationships:**   * **Include** Send invitation   **Business Rules:**   * A sitting request should contain the following information:   + Sitting date (the date that the sitting will happen and should be present or in the future)   + Start time (the time that the sitting will begin)   + End time (the time that the sitting will end)   + Sitting address (the address of the sitting)   + Min age of children (the minimum age of the selected children)   + Total number of children (the number of selected children)   + Total price (the charge fee) * The parent must provide the following information:   + Sitting date   + Start time   + End time   + Min age of children   + Total number of children * Sitting address will be fill auto by the system and the parent cannot modify this information when create a request. * Price for the request is calculate based on the given price of the system and the start time and end time of the sitting request. The parent cannot modify price for the request. * Based on the given information, system will match babysitters with the sitting request. * The parent must provide a valid credit card to create a sitting request. * A sitting request can only be created when the parent send an invitation to a babysitter and only on the first invitation. * System will charge the parent with the total price amount when a sitting request is created. * The newly created sitting request will have the status “Pending” * The newly created invitations will have the status “Pending” | | | |

Table 16 - Use case PR.01 - <Parent> Create babysitting request

###### <Parent> Get babysitting request list

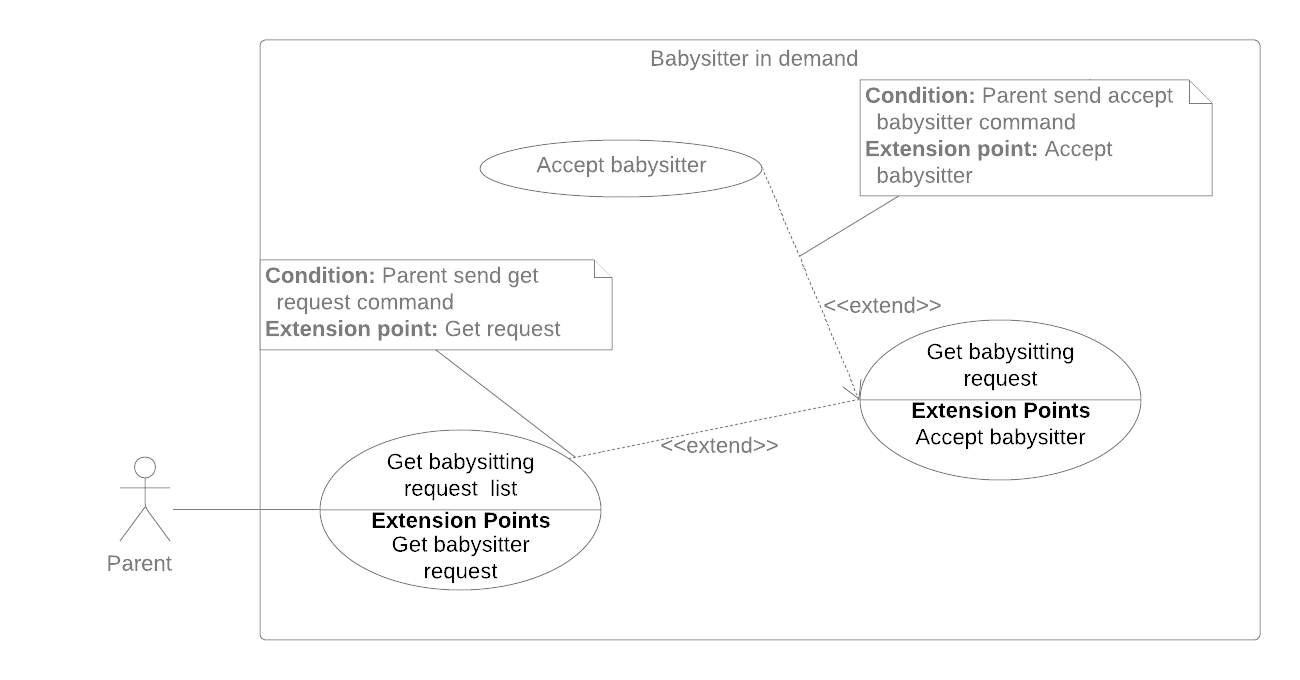


Figure 14 - <Parent> Get babysitting request list

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **USE CASE – UC\_PR.02** | | | | |
| **Use Case No.** | UC\_PR.02 | **Use Case Version** | | 2.0 |
| **Use Case Name** | Get babysitting request list | | | |
| **Author** | DuongPH | | | |
| **Date** | 30/09/2019 | **Priority** | Normal | |
| **Actor:**   * Parent   **Summary:**   * This use case allows parents to get babysitting request list.   **Goal:**   * To help parent manage their requests.   **Triggers:**   * Parent send command to get babysitting request list.   **Preconditions:**   * User must login to the system with role Parent.   **Post Conditions:**   * **Success:**    + System return a list of babysitting requests created by the parent. * **Fail**:   + System shows error message.   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | **Step** | **Actor Action** | **System Response** | | 1 | Parent send command to get the babysitting request list | System return a list of babysitting requests created by the parent.  **[Alternative 1, 2]** |   **Alternative Scenario:**  **Alternative 1:**   |  |  |  | | --- | --- | --- | | **Step** | **Cause** | **System Response** | | 1 | Parent didn’t create any sitting request | System notify the parent about the issues. |   **Alternative 2:**   |  |  |  | | --- | --- | --- | | **Step** | **Cause** | **System Response** | | 1 | Parent don’t have any request with status “Pending”, “Confirmed”, “Ongoing”. | System notify the parent about the issues. |   **Exceptions: N/A**  **Relationships:**   * **extension point** Get babysitting request   **Business Rules:**   * The babysitting request list should be in one of following status: “Pending”, “Confirmed”, “Ongoing”. * The list should be sort by date and time ascending. | | | | |

Table 17 - Use case PR.02 - <Parent> Get babysitting request list

###### <Parent> Get babysitting request

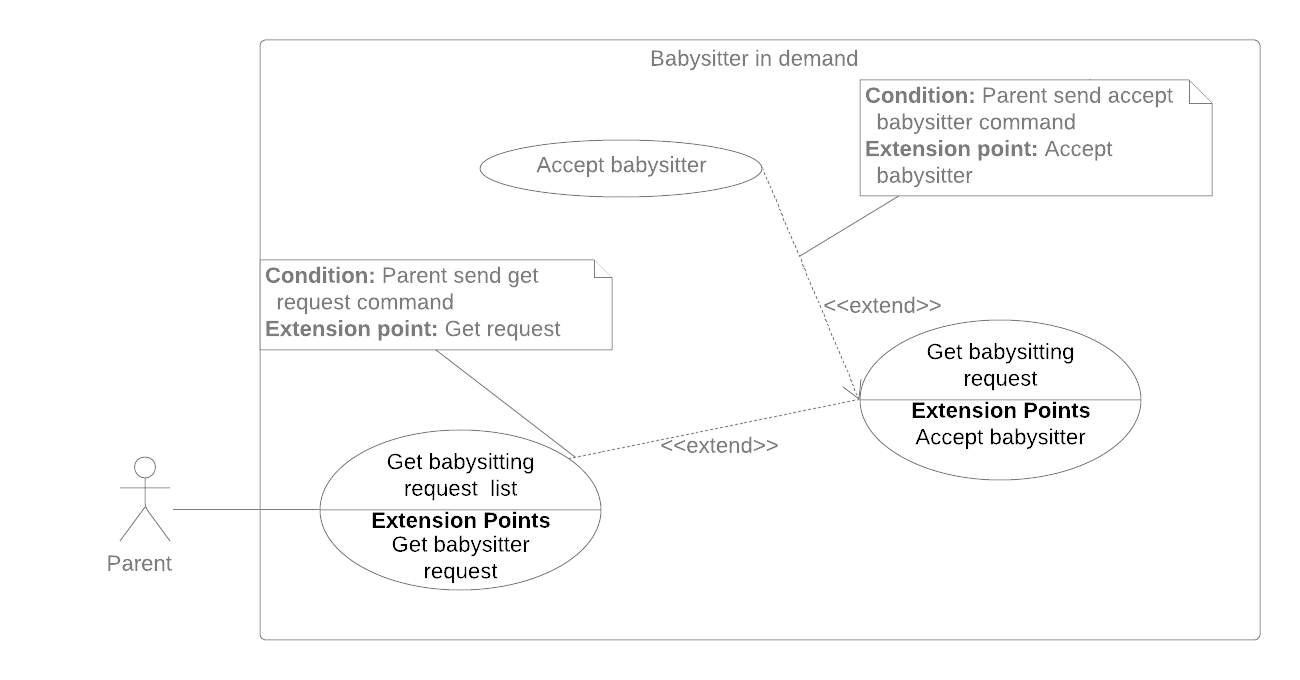


Figure 15 - <Parent> Get babysitting request

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **USE CASE – UC\_PR.03** | | | | |
| **Use Case No.** | UC\_PR.03 | **Use Case Version** | | 2.0 |
| **Use Case Name** | Get babysitting request detail | | | |
| **Author** | DongHN | | | |
| **Date** | 30/09/2019 | **Priority** | Normal | |
| **Actor:**   * Parent   **Summary:**   * This use case allows parent to get babysitting request detail.   **Goal:**   * System returns detail for user.   **Triggers:**   * Parent sends command to get babysitting request detail.   **Preconditions:**   * User must log into the system with role Parent. * The sitting request is created before by the logged account.   **Post Conditions:**   * **Success:**   + System shows sitting request detail. * **Fail:**   + System shows error message.   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | **Step** | **Actor Action** | **System Response** | | 1 | Parent sends command to get babysitting request.  **[Exception 1]** | System returns sitting request detail on screen |   **Alternative Scenario:**  N/A  **Exceptions**:   |  |  |  | | --- | --- | --- | | **No** | **Cause** | **System Response** | | 1 | Failed to get babysitting request. | System shows error message. |   **Relationships:**   * **extend** Get babysitting request list * **extension point** Accept babysitter   **Business Rules:**   * When tap on the sitting request, system will get detail follow sitting id. * The sitting detail is including this information:   + Date   + Start time – End time   + Address   + Price * When the sitting status is PENDING:   + Parent can cancel the request and get a refund from system. System will change status of the sitting to CANCELED when Parent send command to cancel.   + Parent can send command to view “List babysitter” to invite for the sitting.   + List babysitter invited is showed (if babysitter accepted the invitation from Parent for that sitting). Parent can send command to assign that babysitter for the sitting. Status of the sitting will change into CONFIRMED when Parent accept the babysitter. * When the sitting status is CONFIRMED:   + Babysitter information is showed.   + Parent can send command to cancel the request. System will change status of the sitting to CANCELED when Parent send cancel command.   + When it’s sitting time, Parent can send command to check-in assigned babysitter, this will trigger to open camera on Parent’s phone and show QR on Babysitter’s phone for check-in action. When the check-in happen, status of the sitting will change into ONGOING. * When the sitting status is ONGOING:   + After 30 minutes from start time, Parent can send command to checkout assigned babysitter, this will trigger to open camera on Parent’s phone and show QR on Babysitter’s phone for checkout action. When the checkout happen, status of the sitting will change into DONE. | | | | |

Table 18 - Use case PR.03 - <Parent> Get babysitting request

###### <Parent> Accept babysitter

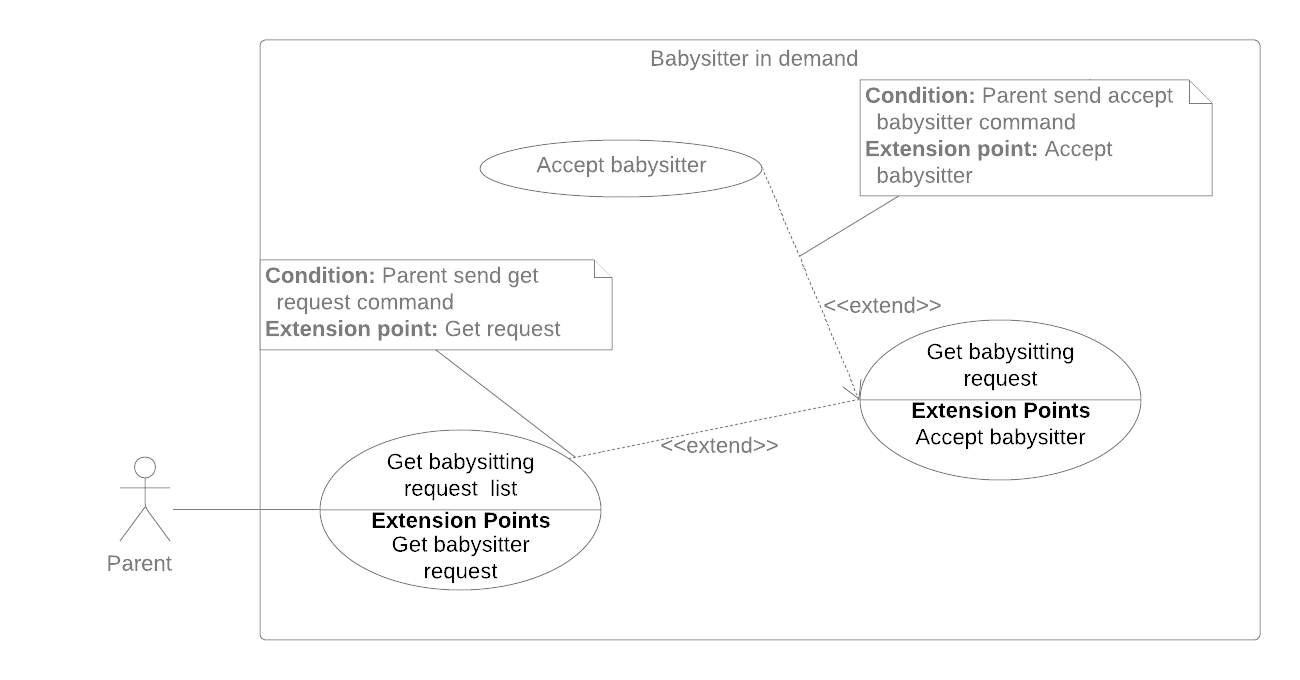


Figure 16 - <Parent> Accept babysitter

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **USE CASE – UC\_PR.04** | | | | |
| **Use Case No.** | UC\_PR.04 | **Use Case Version** | | 2.0 |
| **Use Case Name** | Accept babysitter | | | |
| **Author** | DongHN | | | |
| **Date** | 30/09/2019 | **Priority** | High | |
| **Actor:**   * Parent   **Summary:**   * This use case allows parents to accept a babysitter for a sitting request.   **Goal:**   * Parent can assign a babysitter for the sitting request.   **Triggers:**   * Parent send command to accept babysitter   **Preconditions:**   * User must log into the system with account has role Parent. * The sitting request’s status is PENDING. * The invitation of that request sent to the babysitter have been accepted by babysitter.   **Post Conditions:**   * **Success**: * System assign babysitter for the sitting request. * The sitting request’s status is change into CONFIRMED. * **Fail**: * System show error message.   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | **Step** | **Actor Action** | **System Response** | | 1 | Parent send command to accept a babysitter for a sitting request | System check if the babysitter is available for the parent request.  **[Exception 1]**  If the babysitter is available:  System accept the babysitter as the worker for the parent request  System change the request status to “Confirmed”  System change the invitation of the babysitter to “Confirmed”  System change overlap invitations sent to this babysitter to “Overlap”  System change other invitations of this request to “Expired”  System notify the parent and the babysitter about the success.  **[Exception 2]** |   **Alternative Scenario:**  N/A  **Exceptions**:   |  |  |  | | --- | --- | --- | | **No** | **Cause** | **System Response** | | 1 | The parent request is overlapping with another request accepted by the babysitter | System notify the parent about the problem. | | 2 | Unexpected error happened | System rollback actions, write logs and notify the parent about the problem. |   **Relationships:**   * **extend** Get babysitting request   **Business Rules:**   * Parent cannot accept two babysitters for the same sitting request. * Parent cannot deny a babysitter. * When Parent accepted a babysitter for their sitting request, all other invitations of that sitting request should be change to “Expired” status except for the invitation of the accepted babysitter will be change to “Confirmed” * Any invitations sent to the accepted babysitter with the sitting date and time overlapping with the accepted sitting request should be change to “Overlap” * The accepted babysitter should be notifying about the action. * The sitting request status should be changed to “Confirmed” | | | | |

Table 19 - Use case PR.04 - <Parent> Accept babysitter

###### <Parent> Add people to trust circle

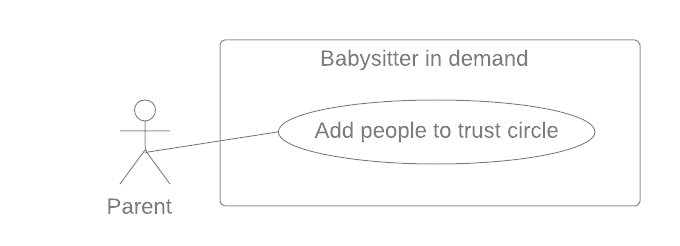


Figure 17 - <Parent> Add people to trust circle

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **USE CASE – UC\_PR.05** | | | | |
| **Use Case No.** | UC\_PR.05 | **Use Case Version** | | 2.0 |
| **Use Case Name** | Add people to trust circle | | | |
| **Author** | DuongPH | | | |
| **Date** | 28/09/2019 | **Priority** | Normal | |
| **Actor:**   * Parent   **Summary:**   * This use case allow parent to add people (other parent in the system) to their trust circle   **Goal:**   * To help parent manage their trust circle   **Triggers:**   * Parent sends command to add more people to trust circle   **Preconditions:**   * User must have logged in to the system and authorized with parent role   **Post Conditions:**   * **Success**:   + Successfully add other people to the circle * **Fail**:   + System notify the error   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | **Step** | **Actor Action** | **System Response** | | 1 | Parent sends command to add more people to trust circle | System ask the parent to search for the other parent they want to add. | | 2 | Parent input the parent code of the other parent and search | System find the other parent by the code.  System return the result to the parent. | | 3 | Parent sends Add to circle command | System add the other parent to the circle of this parent.  **[Exception 1]**  System notify parent that the circle was created successfully | | | | | |
| **Alternative Scenario:** N/A  **Exceptions**:   |  |  |  | | --- | --- | --- | | **No** | **Cause** | **System Response** | | 1 | The other parent is already in this parent circle | System notify this parent about the issue. |   **Relationships:** N/A  **Business Rules:**   * Trust circle consist of the owner parent and any other parent they added. * Every parent owns a trust circle. * Trust circle help the parent find more reliable babysitter based on another parent in the circle previous sitting request. * If a babysitter work for a parent who is in the trust circle, they are more likely to get recommended to the parent. | | | | |

Table 20 - Use case PR.05 - <Parent> Add people to trust circle

###### <Parent> Get request history

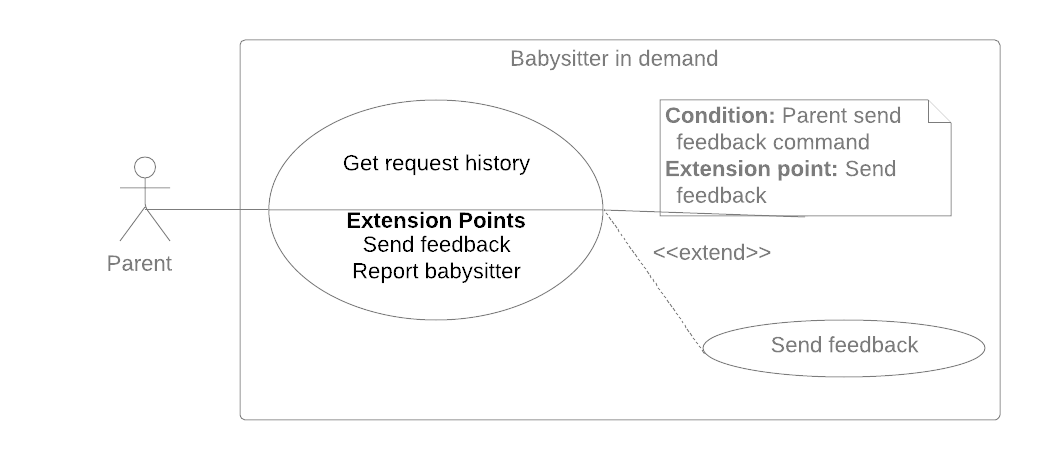


Figure 18 - <Parent> Get request history

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **USE CASE – UC\_PR.06** | | | | |
| **Use Case No.** | UC\_PR.06 | **Use Case Version** | | 2.0 |
| **Use Case Name** | Get request history | | | |
| **Author** | PhucPT | | | |
| **Date** | 30/9/2019 | **Priority** | Normal | |
| **Actor:**   * Parent   **Summary:**   * This use case allows parents to get request history.   **Goal:**   * System return the parent request history   **Triggers:**   * Parent send get request history command.   **Preconditions:**   * User must login into the system with role Parent   **Post Conditions:**   * **Success**: * System return parent request history. * **Fail**: * System shows error messages.   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | **Step** | **Actor Action** | **System Response** | | 1 | Parent sends command to get request history  **[Exception 1]** | System return parent request history | | | | | |
| **Alternative Scenario:** N/A  **Exceptions:**   |  |  |  | | --- | --- | --- | | **No** | **Cause** | **System Response** | | 1 | Error when get request history | System show error message | | | | | |
| **Relationships:**   * **Send feedback** extends **get request history**   **Business Rules:**   * System displays list request history in the order of the latest date. * The history list stores requests with status: * DONE * CANCELED * DONE\_UNCONFIRMED * DONE\_BY\_NEW\_START * SITTER NOT CHECKING * EXPIRED | | | | |

Table 21 - Use case PR.06 - <Parent> Get request history

###### <Parent> Get profile

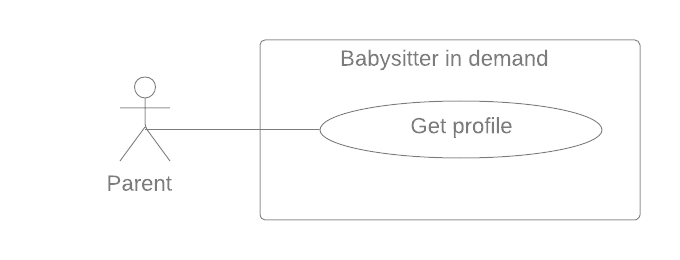


Figure 19 - <Parent> Get profile

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **USE CASE – UC\_PR.07** | | | | |
| **Use Case No.** | UC\_ PR.07 | **Use Case Version** | | 2.0 |
| **Use Case Name** | Get profile | | | |
| **Author** | DongHN | | | |
| **Date** | 27/09/2019 | **Priority** | Normal | |
| **Actor:**   * Parent   **Summary:**   * This use case allows Parent to get account’s profile.   **Goal:**   * System returns parent profile.   **Triggers:**   * Babysitter sends command to get parent profile.   **Preconditions:**   * User must login to the system with the role “Parent”. * Information has been checked by staff.   **Post Conditions:**   * **Success:**    + System returns parent profile. * **Fail**:   + Show error message.   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | **Step** | **Actor Action** | **System Response** | | 1 | Parent sends command to get parent profile. | System returns parent profile detail by Id.  **[Exceptions 1]** | | | | | |
| **Alternative Scenario:** N/A  **Exceptions:**   |  |  |  | | --- | --- | --- | | **No** | **Cause** | **System Response** | | 1 | Error when get parent profile | System show error message |   **Relationships:** N/A  **Business Rules:**   * Babysitter profile detail has the following:   + - Profile picture     - First name     - Last Name     - Birth Day     - Gender     - Address     - Number of children     - Name of children     - Age of child * All information cannot be modified by parent. * Personal information is only changed when verified by BSD staff. | | | | |

Table 22 - Use case PR.07 - <Parent> Get profile

#### <Payment> Pay

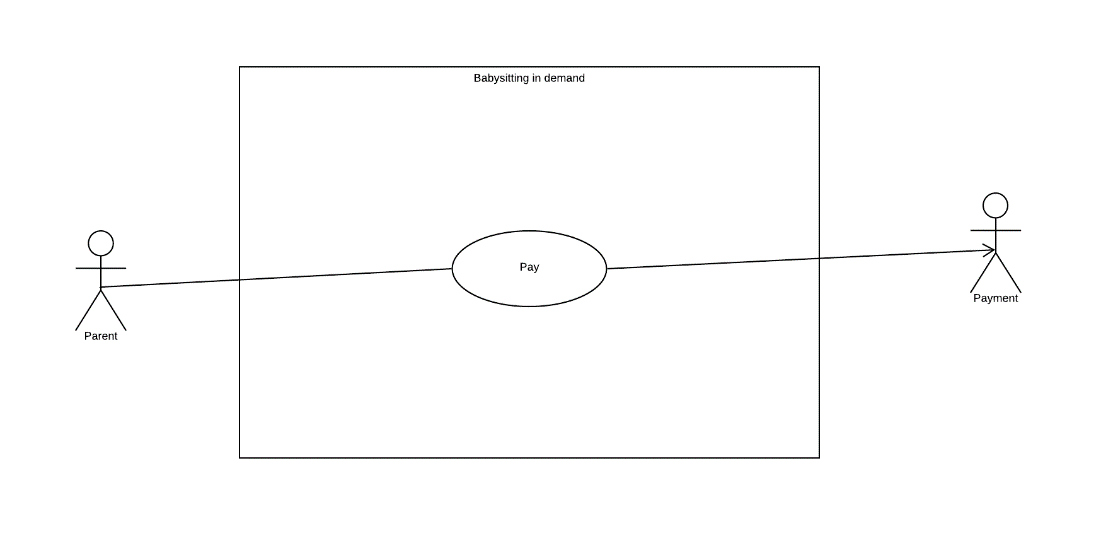


Figure 20 - <Payment> Pay

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **USE CASE – UC\_PM.01** | | | | |
| **Use Case No.** | UC\_PM.01 | **Use Case Version** | | 2.0 |
| **Use Case Name** | Pay | | | |
| **Author** | PhucPT | | | |
| **Date** | 27/09/2019 | **Priority** | High | |
| **Actor:**   * Payment   **Summary:**   * This use case allows the payment to charge the user credit card   **Goal:**   * Payment charge the user’s credit card and save the record to the database   **Triggers:**   * Payment send charge command   **Preconditions:**   * User must have credit card token registered to their account   **Post Conditions:**   * **Success**:   + System record payment transaction of the payment with babysitter * **Fail**: * System record error message   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | **Step** | **Actor action** | **System Response** | | 1 | Payment send command to make a transaction to the user card | System makes transaction to the use card  **[Exception 1]**  System record the transaction to the database | | | | | |
| **Alternative Scenario:** N/A  **Exceptions**:   |  |  |  | | --- | --- | --- | | **No** | **Cause** | **System Response** | | 1 | The user’s card is expired | System stop and rollback the transaction. | | | | | |
| **Relationships:** N/A  **Business Rules:**   * System check if the user has credit card token * System request card information from 3rd payment (Stripe) to verify the card status * The transaction will be canceled if there’s any error * System send request to charge the user credit card * If Success 3rd payment will send back the charge information and system will record to the database | | | | |

Table 23 - Use case PM.01 - <Payment> Pay

###### <Payment> Refund

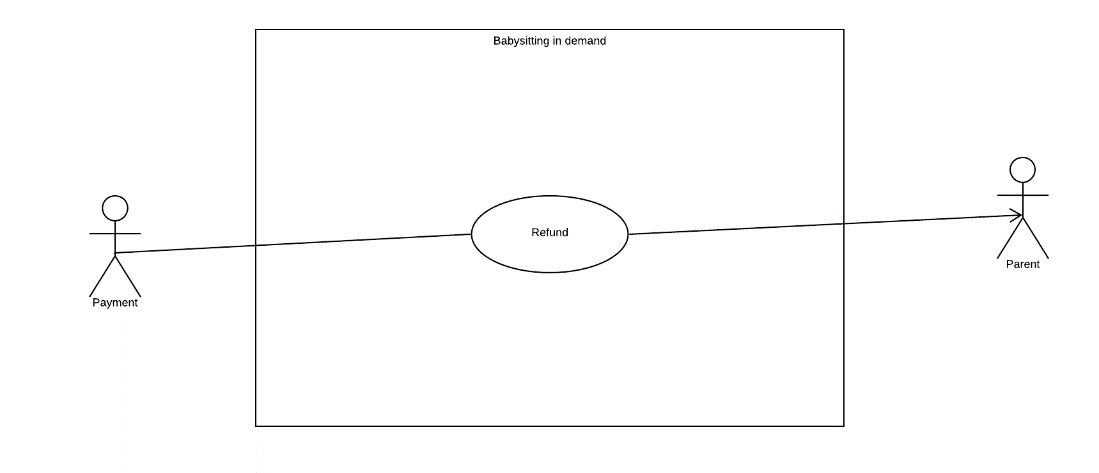


Figure 21 - <Payment> Refund

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **USE CASE – UC\_PM.02** | | | | | |
| **Use Case No.** | UC\_PM.02 | **Use Case Version** | | 2.0 | |
| **Use Case Name** | Refund | | | | |
| **Author** | PhucPT | | | | |
| **Date** | 27/09/2019 | **Priority** | High | |
| **Actor:**   * Payment   **Summary:**   * This use case allows the payment to refund the user payment for the sitting request   **Goal:**   * Refund the Charge for the sitting request with refund fee that was set in the system configuration   **Triggers:**   * Payment send refund command   **Preconditions:**   * User must have credit card token registered to their account * The sitting request mush have already been charged   **Post Conditions:**   * **Success**: * System record payment transaction of the refund * **Fail**: * System record error message   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | **Step** | **Actor action** | **System Response** | | 1 | Payment send command to refund a transaction to the user card | System makes transaction to the use card  **[Exception 1]**  System record the transaction to the database | | | | | | |
| **Alternative Scenario:** N/A  **Exceptions**:   |  |  |  | | --- | --- | --- | | **No** | **Cause** | **System Response** | | 1 | The user’s card is expired | System stop and rollback the transaction. | | | | | | |
| **Relationships:** N/A  **Business Rules:**   * System check if the user has already charged for the sitting request * System request refund to 3rd payment (Stripe) to make refund with the user credit card * System send request to charge the user credit card * If Success 3rd payment will send back the refund information and system will record to the database | | | | | |

Table 24 - Use case PM.02 - <Payment> Refund

#### <System Handler> Overview Use Case

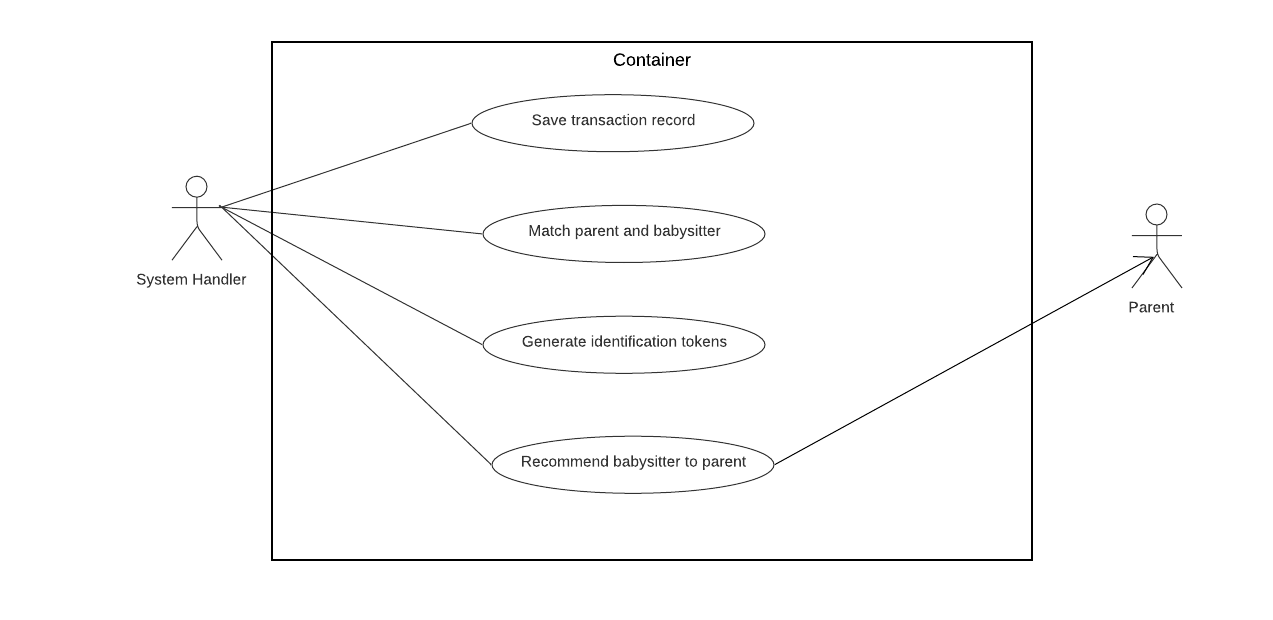


Figure 22 - <System Handler> Overview Use Case



##### <System Handler > Generate identification tokens

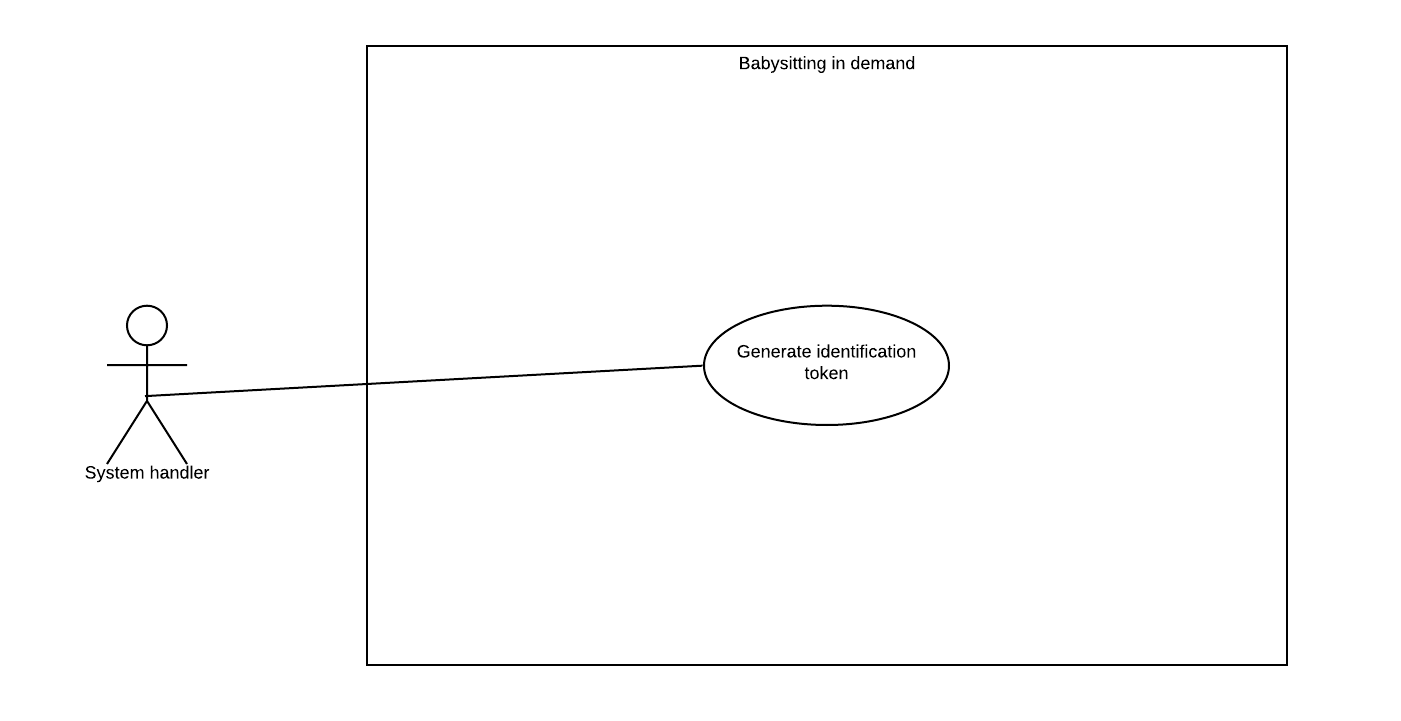


Figure 23 - <System Handler > Generate identification tokens

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **USE CASE – UC\_SH.01** | | | | |
| **Use Case No.** | UC\_SH.01 | **Use Case Version** | | 2.0 |
| **Use Case Name** | Generate identification token | | | |
| **Author** | PhucPT | | | |
| **Date** | 27/09/2019 | **Priority** | High | |
| **Actor:**   * System Handler   **Summary:**   * This use case allows the system handler to generate identification token to verify babysitter when they login   **Goal:**   * System assign Authenticate token to the first-time logging user   **Triggers:**   * System handler send command to generate identification token   **Preconditions:**   * User must be babysitter and their account haven’t been logged into the system before   **Post Conditions:**   * **Success**: * System record the token has been assigned to the user * **Fail**:   + System notifies error message   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | **Step** | **Actor action** | **System Response** | | 1 | System handler sends command to generate token | System will generate token as a QR code or key  **[Exception 1]**  The token then assigned to the newly created account | | | | | |
| **Alternative Scenario:** N/A  **Exceptions**:   |  |  |  | | --- | --- | --- | | No | Cause | System Response | | 1 | Account haven’t been created | Send Error message |   **Relationships:** N/A  **Business Rules:**   * The token will be generated as key code. User can input the key code to their devices * Babysitter need to use the token to be allowed to access the system * The token is generated base on the RFC 6238 document which provide the TOTP (time base one-time password) the short-live otp to enhance security. * There also generated one more token base on the user device to make sure user can only have access to the system if their use the registered device | | | | |
|  | | | | |

Table 25 - Use case SH.01 - <System Handler > Generate identification tokens

##### 1.2.5.2. <System Handler> Save transaction record

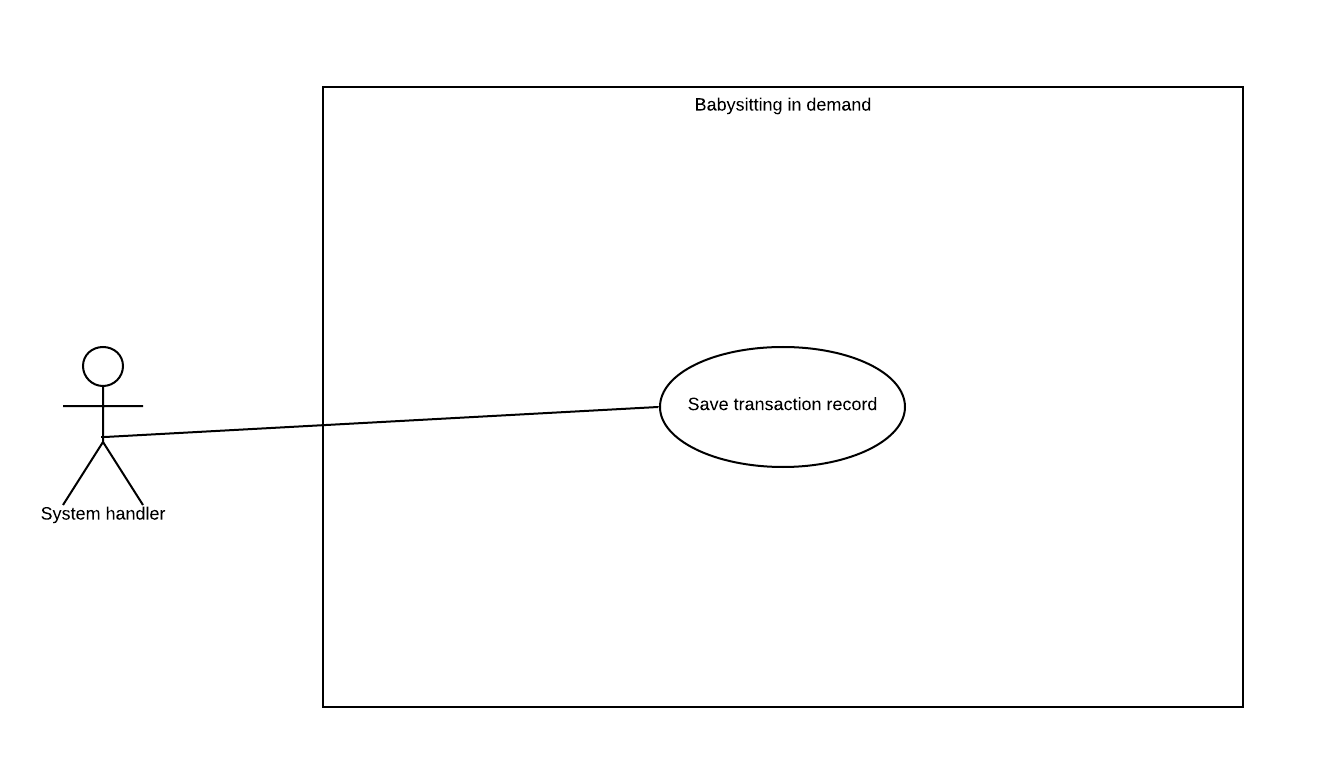


Figure 24 - <System Handler> Save transaction record

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **USE CASE – UC\_SH.02** | | | | |
| **Use Case No.** | UC\_SH.02 | **Use Case Version** | | 2.0 |
| **Use Case Name** | Save transaction record | | | |
| **Author** | PhucPT | | | |
| **Date** | 27/09/2019 | **Priority** | High | |
| **Actor:**   * System Handler   **Summary:**   * This use case allows the system handler to save the transaction of payment   **Goal:**   * To write all of the transaction to the database   **Triggers:**   * System sends command to record the transaction   **Preconditions:**   * Parent has paid for the babysitting * The Refund is successfully requested   **Post Conditions:**   * **Success**:   + System record the transaction * **Fail**:   + System notifies error message   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | **Step** | **Actor action** | **System Response** | | 1 | System handler check if the transaction has been successfully made | **[Exception 1]** | | 2 | System handler send command to record the transaction to the database |  | | | | | |
| **Alternative Scenario:** N/A  **Exceptions**:   |  |  |  | | --- | --- | --- | | **No** | **Cause** | **System Response** | | 1 | Transaction fail | System show error message | | | | | |
| **Relationships:** N/A  **Business Rules:**   * All the transaction will be stop when there is error appear * System only record the transaction:   + When Parent has already paid for the sitting   + When the System have successfully requested the refund from the 3rd payment (Stripe) | | | | |

Table 26 - Use case SH.02 - <System Handler> Save transaction record

##### 1.2.5.3. <System Handler> Matching parent and babysitter

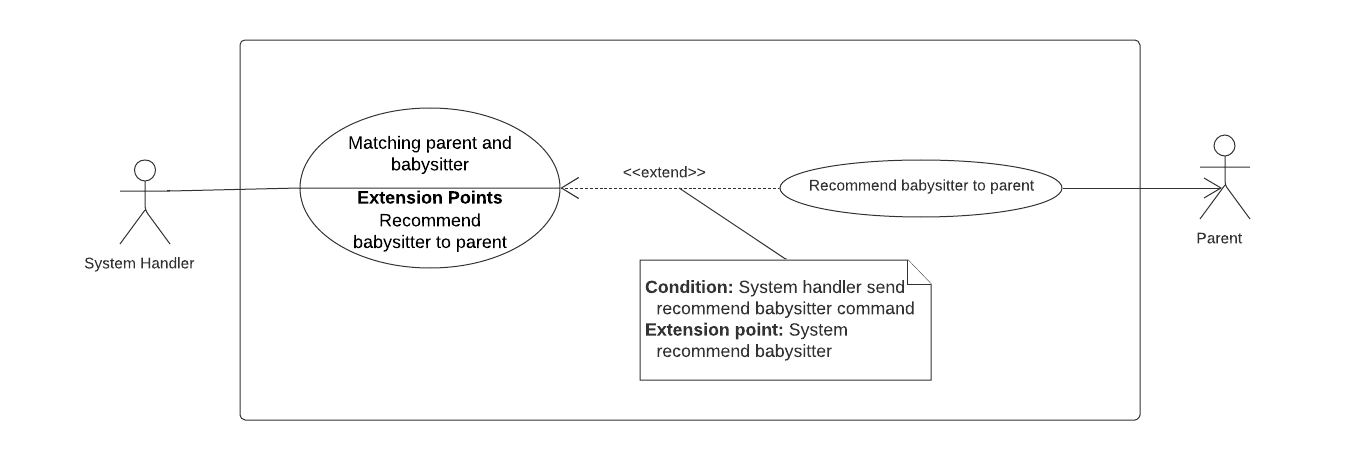


Figure 25 - <System Handler> Matching parent and babysitter

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **USE CASE – UC\_SH.03** | | | | |
| **Use Case No.** | UC\_SH.03 | **Use Case Version** | | 2.0 |
| **Use Case Name** | Matching parent and babysitter | | | |
| **Author** | DuongPH | | | |
| **Date** | 27/09/2019 | **Priority** | High | |
| **Actor:**   * System Handler   **Summary:**   * This use case indicates that the system handler can find match babysitters and parent request.   **Goal:**   * To help parent find babysitter for their sitting request.   **Triggers:**   * Parent find matched babysitter when creating a sitting request. * Parent want view the matched list again.   **Preconditions:**   * There are babysitters in the system   **Post Conditions:**   * **Success**: * System return recommendation list and matched list to parent. * **Fail**: * System notifies error message   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | **Step** | **Actor action** | **System Response** | | 1 | System handler sends command to get the newly created parent’s sitting request. | System finds the corresponding sitting request of the parent.  **[Exception 1]**  System return the sitting request. | | 2 | System handler sends command to start matching babysitters and the parent sitting request. | System finds all babysitters. (including their sitting preferences)  System filter babysitters from the list with the matched sitting preferences to the parent sitting request.  System filter babysitters from the list with an available schedule to the parent sitting request.  System get the distance between the parent sitting address and each babysitter address by using Google API.  System filter babysitters from the list with the distance to parent address < 3km.  System return the matched list. | | 4 | System handler sends command to return the matched list to parent | System sends the matched list to parent.  **[Alternative 1]** | | | | | |
| **Alternative Scenario:** N/A  **[Alternative 1]**   |  |  |  | | --- | --- | --- | | **Step** | **Actor action** | **System Response** | | 1 | The matched list is null | System notify the parent about the result. | | | | | |
| **Exceptions**:   |  |  |  | | --- | --- | --- | | **No** | **Cause** | **System Response** | | 1 | There is no sitting request from the parent | System stop the matching | | | | | |
| **Relationships:**   * **Extension point** Recommend babysitter to parent.   **Business Rules:**   * The sitting request must at least contain the following information: * The day of the sitting * The time the sitting will happen * The place the sitting will happen (the parent’s address) * The total number of children in the sitting * Minimum age of children * Match babysitters sitting preferences with the parent request by comparing these following criteria:   + Weekly schedule (against the day of the sitting)   + Work time in the day (against the time the sitting will happen)   + The maximum number of children they can look after (against the number of children in the sitting)   + The minimum age number of children they can look after (against age of each child * Filter babysitters with an available schedule by checking if the babysitter has accepted a sitting request that overlap with the parent request. * Get the distance between the parent sitting address and the address of each babysitter in the matched list (Using Google Distance Matrix API) * Filter babysitters with the distance data < 3km * Only babysitters who pass all above criteria can be recommend to the parent. | | | | |

Table 27 - Use case SH.03 - <System Handler> Matching parent and babysitter

##### 1.2.5.4. <System Handler> Recommend babysitter to parent

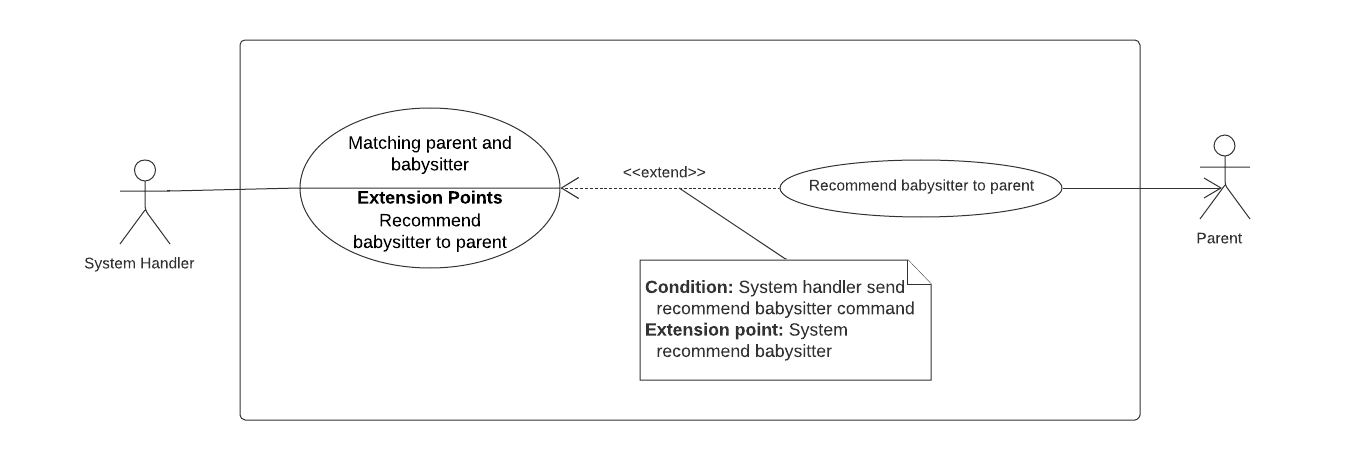


Figure 26 - <System Handler> Recommend babysitter to parent

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **USE CASE – UC\_SH.04** | | | | |
| **Use Case No.** | UC\_SH.04 | **Use Case Version** | | 2.0 |
| **Use Case Name** | Recommend babysitter to parent | | | |
| **Author** | DuongPH | | | |
| **Date** | 27/09/2019 | **Priority** | High | |
| **Actor:**   * System Handler   **Summary:**   * This use case allows the system handler to recommend babysitter suitable to the parent   **Goal:**   * Recommend suitable babysitter to parent   **Triggers:**   * Parent has just created a babysitting request and matching with babysitter is done   **Preconditions:**   * Parent’s babysitting request must match with at least one babysitter   **Post Conditions:**   * **Success**: * System suggest suitable babysitter(s) to parent * **Fail**: * System notifies error message   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | **Step** | **Actor action** | **System Response** | | 1 | System handler sends command to get recommendation list | System get list of matched babysitters.  **[Exception 1]**  System calculate the total score of each matched babysitter in the list.  System sort the list and take the top 5 highest score babysitter.  System return the result. | | 2 | System handler sends command to send the recommendation list to parent. | System sends the recommendation list to parent. | | | | | |
| **Alternative Scenario:** N/A  **Exceptions**:   |  |  |  | | --- | --- | --- | | **No** | **Cause** | **System Response** | | 1 | There is no matched babysitter | System stop recommending. | | | | | |
| **Relationships:** Parent  **Business Rules:**   * Matched result of babysitters is babysitter whose sitting preferences match with the parent’ sitting request preferences:   + Available date (against the day of the sitting)   + Work time in day (against the time the sitting will happen)   + The maximum number of children they can look after (against the number of children in the sitting)   + The minimum age number of children they can look after (against age of each child * Parent’s babysitting request must match with at least two babysitters for the system handler to start the recommending process * System handler should only recommend babysitters who is in the matched list with the parent sitting preferences * Total score of a babysitter is calculated based on 3 criteria:   + If the babysitter has worked for a parent and that parent is in the trust circle of the requesting parent (50% of the total score)   + The babysitter’s rating (40% of the total score)   + The travel distance between the babysitter and the parent (10% of the total score)   + The babysitter rating must have minimum of 5 vote to be consider a valid rating. * The system handler should only show at most top 5 babysitters when recommending | | | | |

Table 28 - Use case SH.04 - <System Handler> Recommend babysitter to parent

#### <Staff> Overview Use Case

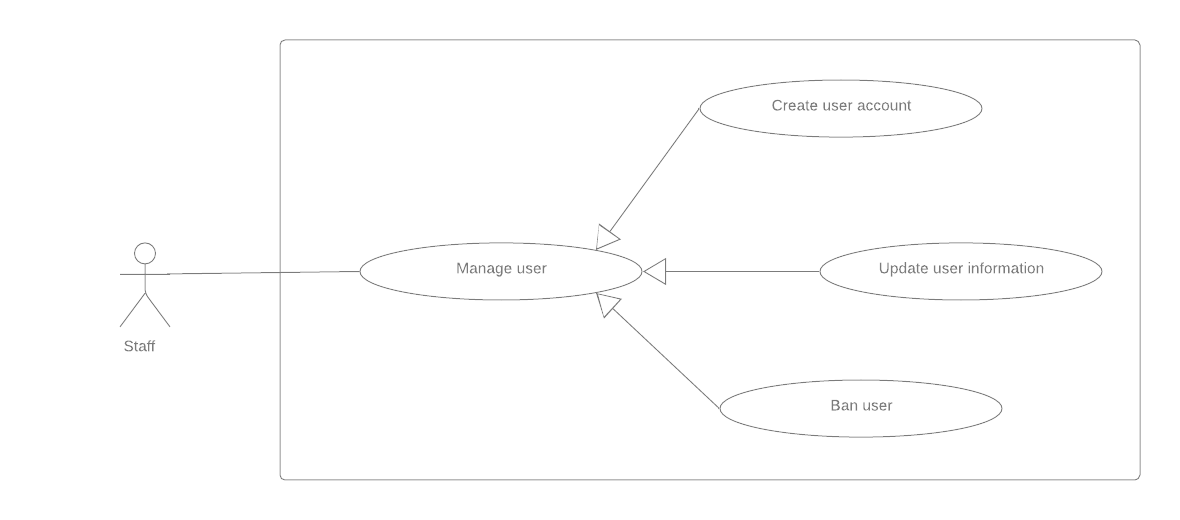


Figure 27 - <Staff> Overview Use Case



##### <Staff> Create user account

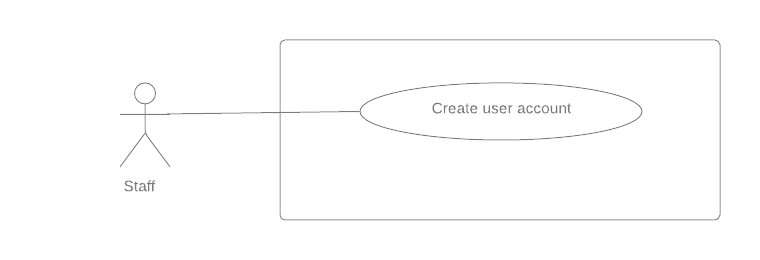


Figure 28 - <Staff> Create user account

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **USE CASE – UC\_ST.01** | | | | |
| **Use Case No.** | UC\_ST.01 | **Use Case Version** | | 2.0 |
| **Use Case Name** | Create user account | | | |
| **Author** | DuongPH | | | |
| **Date** | 27/09/2019 | **Priority** | Normal | |
| **Actor:**   * Staff   **Summary:**   * This use case allow staff to create account for user like parent and babysitter   **Goal:**   * To help staff manage user account   **Triggers:**   * Staff sends command to create user account   **Preconditions:**   * User must have logged in to the system and authorized with staff role   **Post Conditions:**   * **Success**:   + User’s account created * **Fail**: * System notifies error message   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | **Step** | **Actor Action** | **System Response** | | 1 | Staff sends command to create user account | System shows create new account page | | 2 | Staff choose role to create the new account | System shows the corresponding form to the chosen role | | 3 | Staff fill in the form |  | | 4 | Staff sends submit form command | System validates information  **[Exception 1, 2]**  System create new account with given information  System notify staff that the account was created successfully | | | | | |
| **Alternative Scenario:** N/A  **Exceptions**:   |  |  |  | | --- | --- | --- | | **No** | **Cause** | **System Response** | | 1 | Staff input invalid information | System show error message and show the requirement for the correct information to staff | | 2 | User’s phone number existed in the system | System show error message that the phone number is existed in the system | | | | | |
| **Relationships:** N/A  **Business Rules:**   * Sitter account’s register form: * Avatar. * Fullname. * Gender. * Date of birth. * Email. * Phone number. * Address. * Working day. * Working day time. * Working evening time. * Min. age of children. * Max. number of children. * Max. distance. * Parent account’s register form: * Avatar. * Fullname. * Gender. * Date of birth. * Email. * Phone number. * Address. * Number of children. * This use case can only be done by staff and on the web application * There are two roles for staff to choose when create a new account: parent and babysitter * Each role has different input field and requirement * User account created by this use case will has their phone number as username and their first password generate randomly. Therefore, phone number cannot be duplicate in the system * The created account will have status as “Active” indicate that the account is active | | | | |

Table 29 - Use case ST.01 - <Staff> Create user account

##### <Staff> Update user information

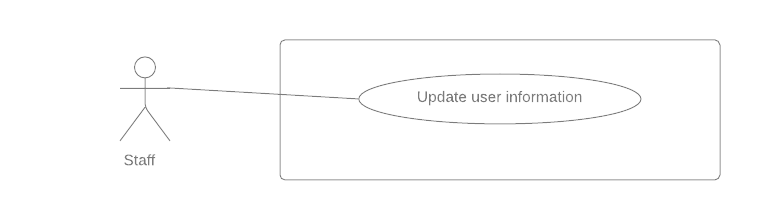


Figure 29 - <Staff> Update user information

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **USE CASE – UC\_ST.02** | | | | |
| **Use Case No.** | UC\_ST.02 | **Use Case Version** | | 2.0 |
| **Use Case Name** | Update user information | | | |
| **Author** | DuongPH | | | |
| **Date** | 27/09/2019 | **Priority** | Normal | |
| **Actor:**   * Staff   **Summary:**   * This use case allow staff to update information of user like parent or babysitter   **Goal:**   * To help staff manage user account   **Triggers:**   * Staff sends command to update user information   **Preconditions:**   * User must have logged in to the system and authorized with staff role   **Post Conditions:**   * **Success**:   + User’s account information updated * **Fail**: * System notifies error message   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | **Step** | **Actor Action** | **System Response** | | 1 | Staff sends command to update a user information | System check the user account role  System shows the corresponding user information page based on account role | | 2 | Staff change user information |  | | 3 | Staff sends submit form command | System validates information  **[Exception 1, 2]**  System update the account with given information  System notify staff that the account was updated successfully | | | | | |
| **Alternative Scenario:** N/A  **Exceptions**:   |  |  |  | | --- | --- | --- | | **No** | **Cause** | **System Response** | | 1 | Staff input invalid information | System show error message and show the requirement for the correct information to staff | | 2 | User’s new phone number existed in the system | System show error message that the new phone number is existed in the system | | | | | |
| **Relationships:** N/A  **Business Rules:**   * Sitter account’s register form: * Email. * Phone number. * Address. * Working day. * Working day time. * Working evening time. * Parent account’s profile form: * Email. * Phone number. * Address. * Number of children. * If all information is valid, system checks if phone number, email has already existed in system or not. * This use case can only be done by staff and on the web application * When staff choose a user account to update, system will check if that account role is parent or babysitter and redirect to corresponding page * Each role has different input field and requirement * If the phone number change, system should check if the new phone number is duplicate with any other account in the system | | | | |

Table 30 - Use case ST.02 - <Staff> Update user information

##### <Staff> Ban user

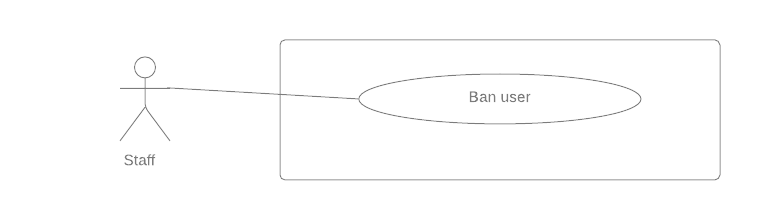


Figure 30 - <Staff> Ban user

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **USE CASE – UC\_ST.03** | | | | |
| **Use Case No.** | UC\_ST.03 | **Use Case Version** | | 2.0 |
| **Use Case Name** | Ban user | | | |
| **Author** | DuongPH | | | |
| **Date** | 27/09/2019 | **Priority** | Normal | |
| **Actor:**   * Staff   **Summary:**   * This use case allow staff to ban user like parent or babysitter   **Goal:**   * To help staff manage user account   **Triggers:**   * Staff sends command to ban user account   **Preconditions:**   * User must have logged in to the system and authorized with staff role   **Post Conditions:**   * **Success**: * User’s account updated status to “Banned” * **Fail**: * System notifies error message   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | **Step** | **Actor action** | **System Response** | | 1 | Staff sends command to ban a user | System change the user account status to  “Banned”  **[Exception 1]**  System notify staff that the action was successfully | | | | | |
| **Alternative Scenario:** N/A  **Exceptions**: N/A   |  |  |  | | --- | --- | --- | | **No** | **Cause** | **System Response** | | 1 | User account is already been banned | System notify to staff that the account is already been banned | | | | | |
| **Relationships:** N/A  **Business Rules:**   * This use case can only be done by staff and on the web application * A user account is considered banned when it status is “Banned” * A banned account is no longer able to log in to the system | | | | |

Table 31 - Use case ST.03 - <Staff> Ban user

##### <Staff> Edit configuration

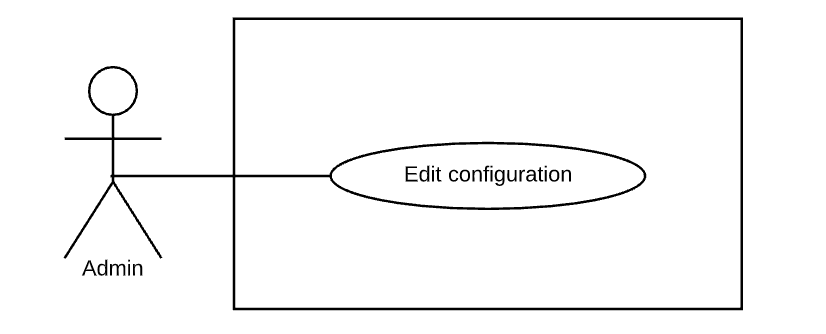


Figure 31 - <Staff> Edit configuration

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **USE CASE – UC\_AD.01** | | | | |
| **Use Case No.** | UC\_AD.01 | **Use Case Version** | | 2.0 |
| **Use Case Name** | Edit configuration | | | |
| **Author** | PhucPT | | | |
| **Date** | 30/09/2019 | **Priority** | Normal | |
| **Actor:**   * Admin   **Summary:**   * This use case allow admin to edit application’s configuration.   **Goal:**   * Admin can edit configuration.   **Triggers:**   * Admin sends command to edit configuration.   **Preconditions:**   * User must have logged into the system with admin role   **Post Conditions:**   * **Success**: * Configuration of the system is edited. * **Fail**: * System notifies error message.   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | **Step** | **Actor action** | **System Response** | | 1 | Admin sends command for editing configuration  **[Alternative 1]** | System show all editable configuration of the system. | | 2 | Admin input required information and sends request to edit configuration. | System display success message | | | | | |
| **Alternative Scenario:**  **[Alternative 1]**   |  |  |  | | --- | --- | --- | | **Step** | **Actor Action** | **System Response** | | 1 | Amin send cancel command | System cancel the editing |   **Exceptions**:   |  |  |  | | --- | --- | --- | | **No** | **Cause** | **System Response** | | 1 | Admin entered invalid parameters. | System show error message on invalid fields. |   **Relationships:** N/A  **Business Rules:**   * This use case can only be done by admin and on the web application * Admin will edit the configuration of the system * Configuration used to edit some parameters which can be changed to fit with business at current or future:   + Price: price per working hour.   + remindBeforeDuration\_0: Second time remind upcoming sitting.   + remindBeforeDuration\_1: First time remind upcoming sitting.   + checkinTimeout: Timeout after sitting’s start time.   + checkoutTimeout: Timeout after sitting’s end time.   + Timezone: Timezone of system.   + maxTravelDistance: Maximum distance can find babysitter to recommend.   + circleWeight: weight to calculate mark of babysitter with a sitting.   + ratingWeight: weight to calculate mark of babysitter with a sitting.   + distanceWeight: weight to calculate mark of babysitter with a sitting   + minimumFeedback: Minimum feedback to display in babysitter profile.   + refundPercentage: Refund percentage when parent cancel a sitting request. | | | | |

Table 32 - Use case ST.04 - <Staff> Edit configuration

## Conceptual Diagram

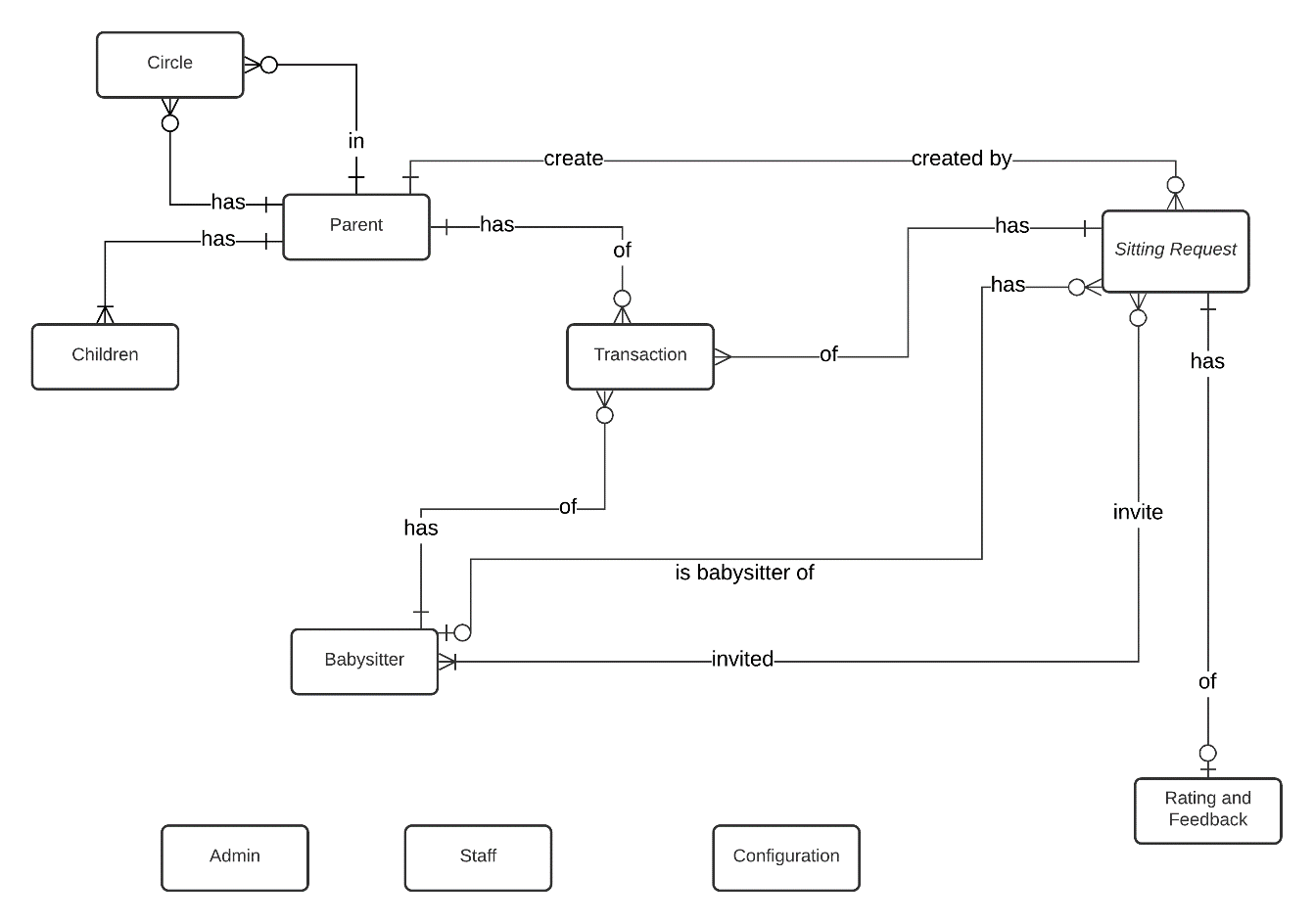


Figure 32 - Conceptual diagram

**Data Dictionary**

|  |  |
| --- | --- |
| **Entity Data Dictionary** | |
| **Entity Name** | **Description** |
| Admin | The administrator of the system |
| Staff | The staff of the system |
| Parent | Represent role Parent in the system |
| Babysitter | Represent role Babysitter in the system |
| Children | Children of the Parent |
| Circle | Represent trust circle of Parent, a Parent can have many other Parent in his/her trust circle and can be in other Parent trust circle as well |
| Sitting request | Represent the unit of work of the system |
| Rating Feedback | Feedback for a sitting request, given by the Parent |
| Transaction | Transaction between the Parent and the Babysitter though the system |
| Configuration | The configuration of the system |

# **Software Design Description**

## Design Overview

This document describes the technical and user interface design of BID System. It includes the architectural design, the component design, the user interface design and the database model:

* The architectural design describes the overall architecture of the system.
* The component design describes all system’s component together with relationships between them. It also explains clearly their purposes, methods within each component and detailed algorithm, pseudo code to implement them.
* User interface design describe all screen’s interface; each screen includes which types of input, output element and used by what action.
* The Database design describes the relationships between entities and details of each entity.

## System Architecture Design

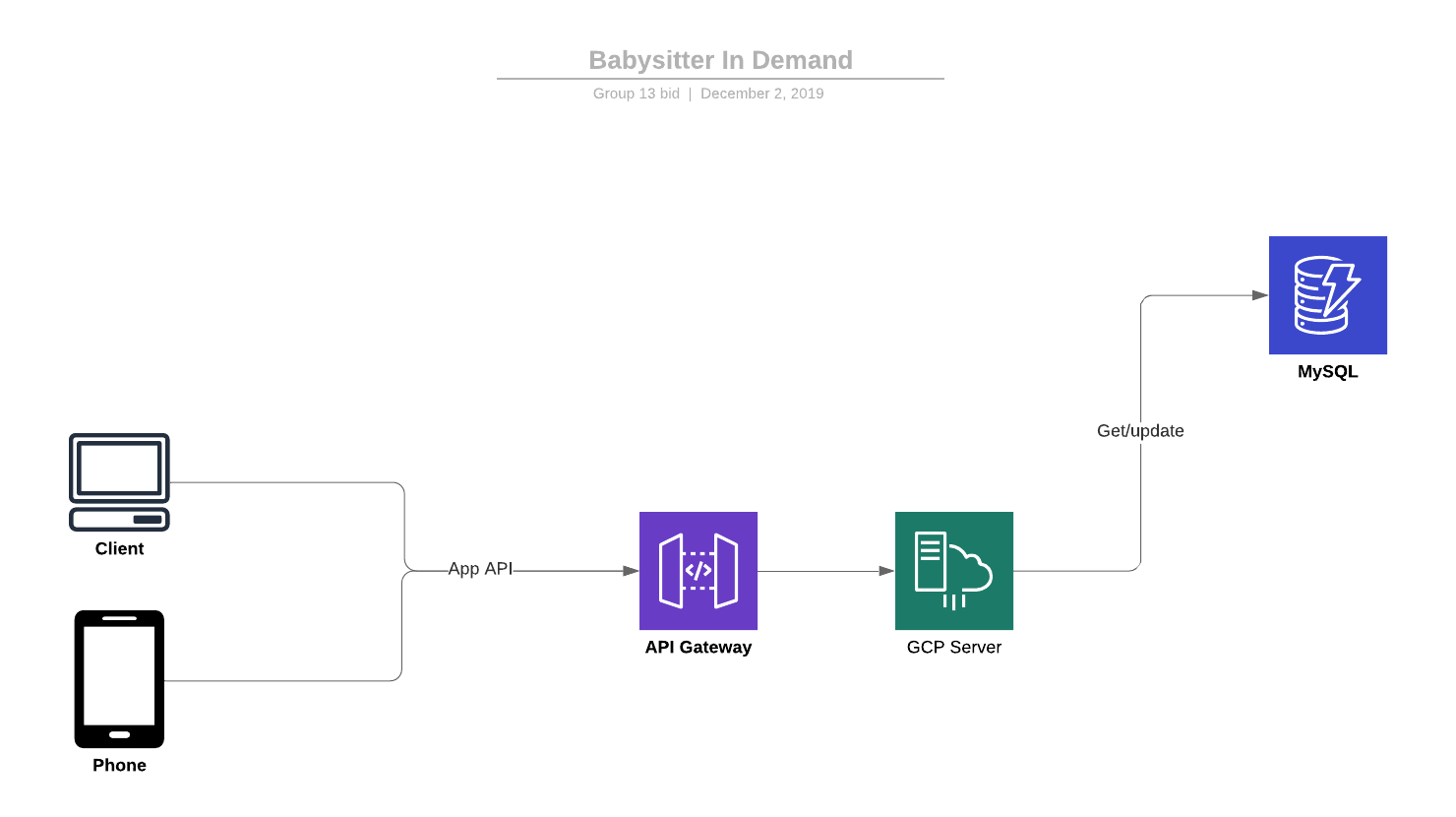


Figure 33 – System architecture design

## Component Diagram

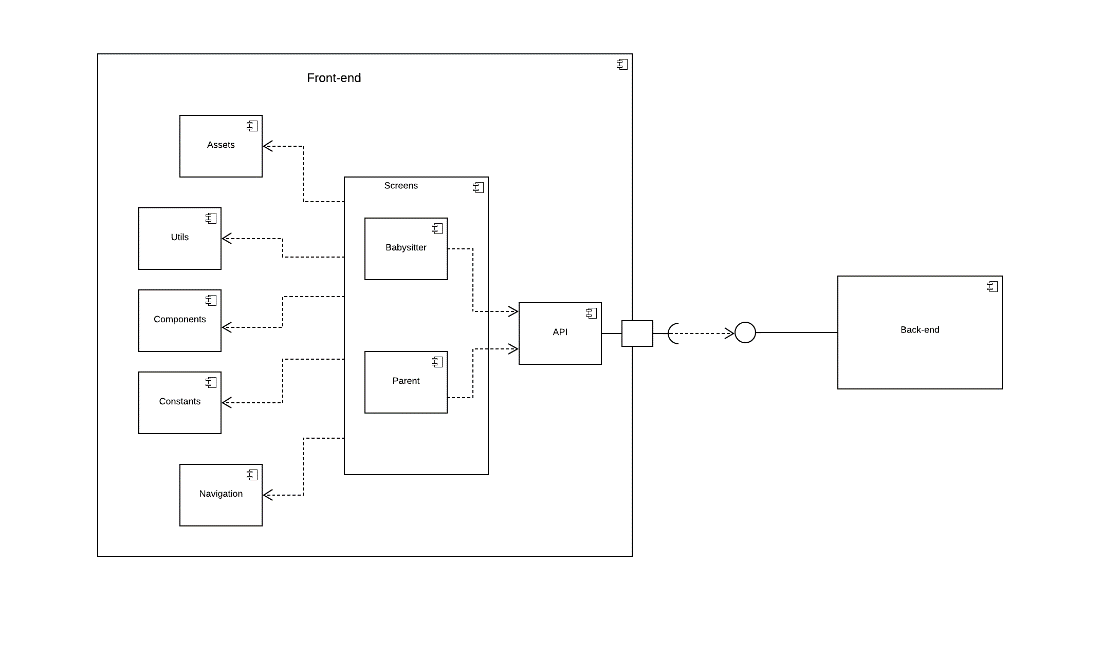


Figure 34 – Front-end component diagram

|  |  |
| --- | --- |
| **Component dictionary: describe components** | |
| **Component Name** | **Description** |
| API | Contains modules related to handle authentication, http request and response, local storage calls and web socket communication calls |
| Screen | Contains all the screens of the application, parent side and sitter side |
| Parent | Contains UI screens of the parent side |
| Sitter | Contains UI screens of the sitter side |
| Assets | Contains static resources for user interface processing: fonts, images, image, layout and utilizations |
| Utils | Contains helper function like local storage, decode, time format parser, loading module. |
| Component | Contains pre-define UI component that can be re-use |
| Constants | Contains all constant used in the system: status of sitting request, color palate, layout |
| Navigation | Contains modules used to navigate between screens |

Table 33 - Front-end component diagram dictionary

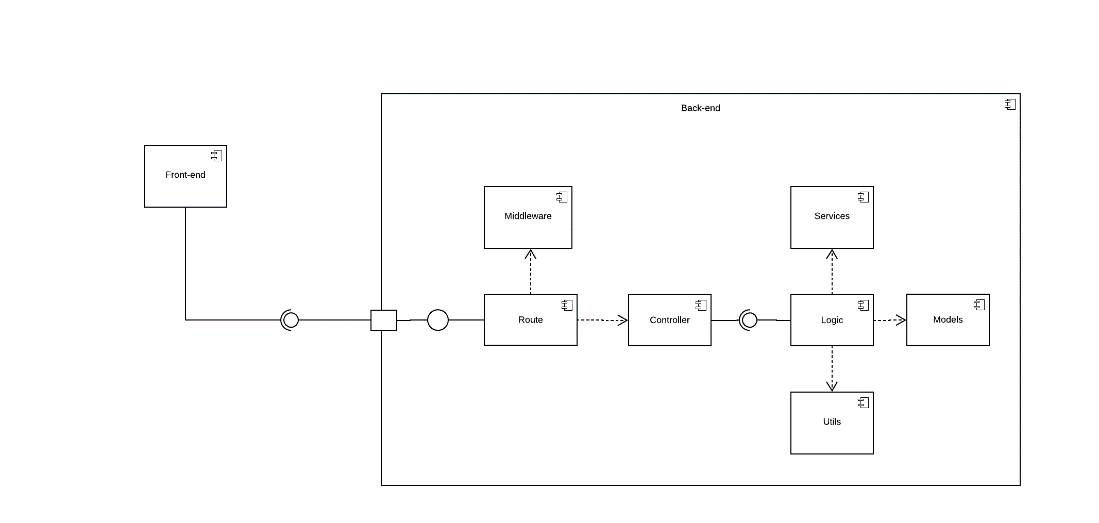


Figure 35 - Back-end component diagram

|  |  |
| --- | --- |
| **Component dictionary: describe components** | |
| **Component Name** | **Description** |
| Route | Contains modules used to navigate HTTP request:  Authentication, user, sitting request, invitation, circle, configuration, feedback, payment, transaction, parent, babysitter, feedback |
| Middleware | Contains modules used to pre-process every request sent to server such as: verify cross-origin, rate limit, verify token, request logging |
| Controller | Provide access to system, contains functions to handle http requests and response: Authentication, user, sitting request, invitation, circle, configuration, feedback, payment, transaction, parent, babysitter, feedback |
| Logic | Contains modules that handle business logic at service layer, handle the interaction between controller and DB: Authentication, user, sitting request, invitation, circle, configuration, feedback, payment, transaction, parent, babysitter, feedback |
| Model | Contains modules related to handle transactions, error response, hash functions, access environment variables, handle interactions between system and database at persistence layer |
| Service | Contains modules that handle accepting, inviting, scheduling, generating token, matching, recommending, notifying. |
| Utils | Contains helper function that handle socket connection, hashing, notifying, generating token. |

Table 34 - Back-end component diagram dictionary

## Detailed Description

### Class Diagram

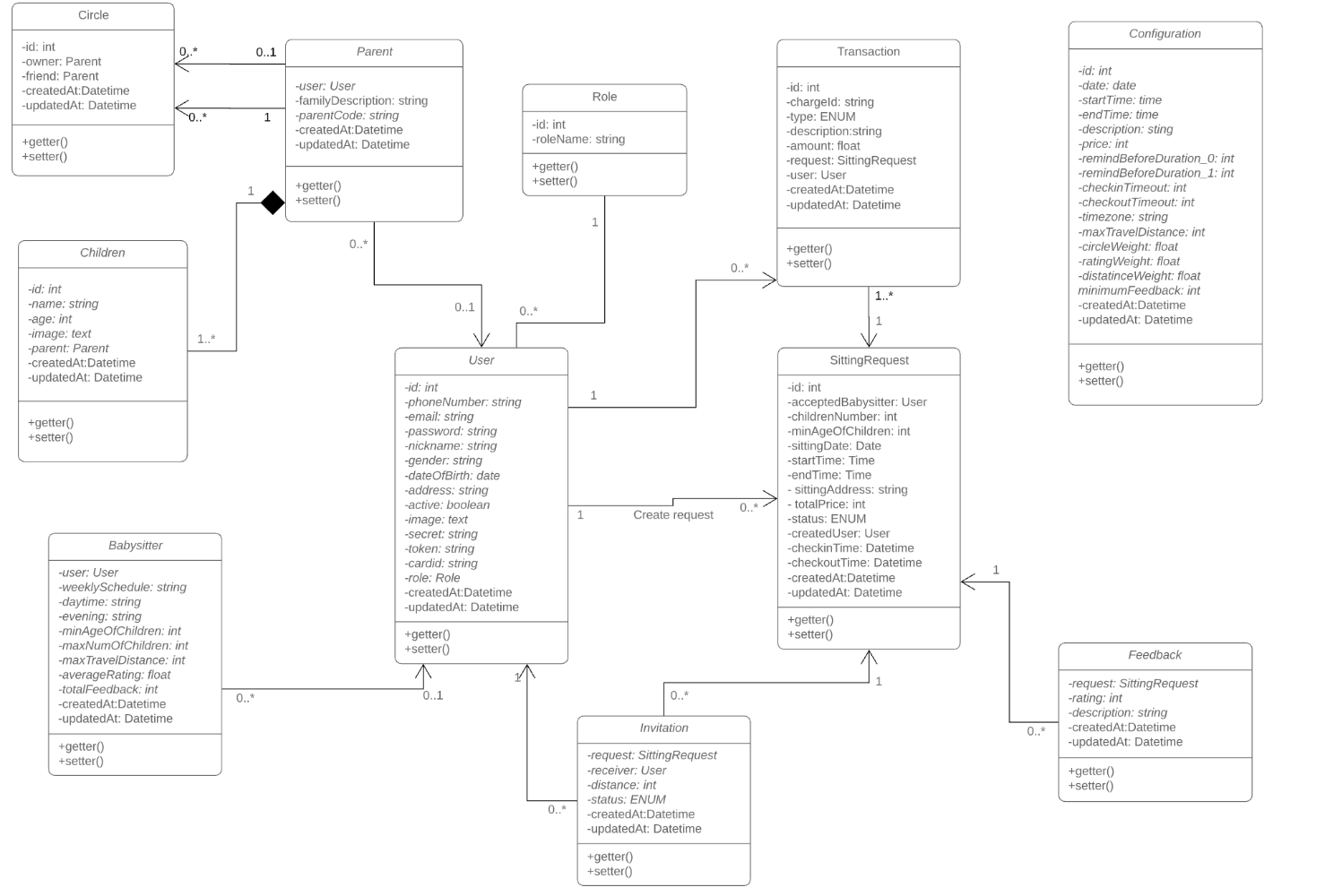


Figure 36 – Class diagram(3)

This class diagram uses the notation elements of UML 2.0

|  |  |  |
| --- | --- | --- |
| **No.** | **Class Name** | **Description** |
| 1 | User | Contain account information include phone number, address, email, full name... |
| 2 | Parent | Contain parent account information include family description, parent code… |
| 3 | Children | Contain children information include name, age, image… |
| 4 | Circle | Contain circle information include owner, friends… |
| 5 | Babysitter | Contain babysitter information include worktime, requirement… |
| 6 | Role | Contain role information of the system. |
| 7 | Sitting Request | Contain sitting request information include address, price, sitting time, … |
| 8 | Feedback | Contain feedback information for a sitting request include rating, description. |
| 9 | Invitation | Contain invitation information for a sitting request include receiver, status. |
| 10 | Configuration | Contain system’s configuration include price per hour, time zone, max travel distance, … |
| 11 | Transaction | Contain transaction information of a sitting request include bill id, type, amount, … |
|  |  |  |

Table 35 – Class Dictionary

### Interactive Diagram

#### <Guest> Login

Summary: This diagram shows the process of verifying the user

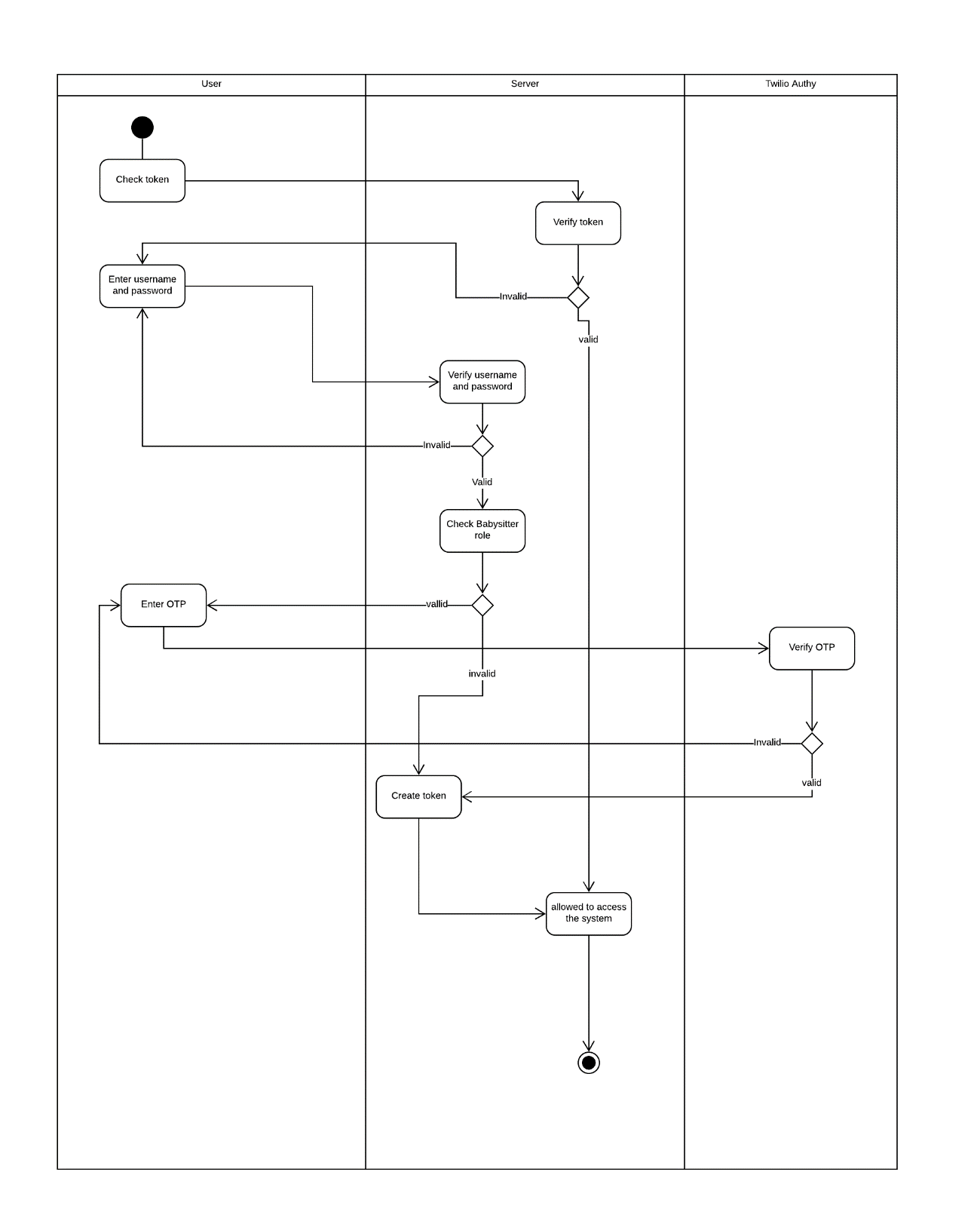


Figure 37 - <Guest> Login

#### <Babysitter> Answer invitation

Summary: This diagram shows the process of babysitter answer invitation from parent

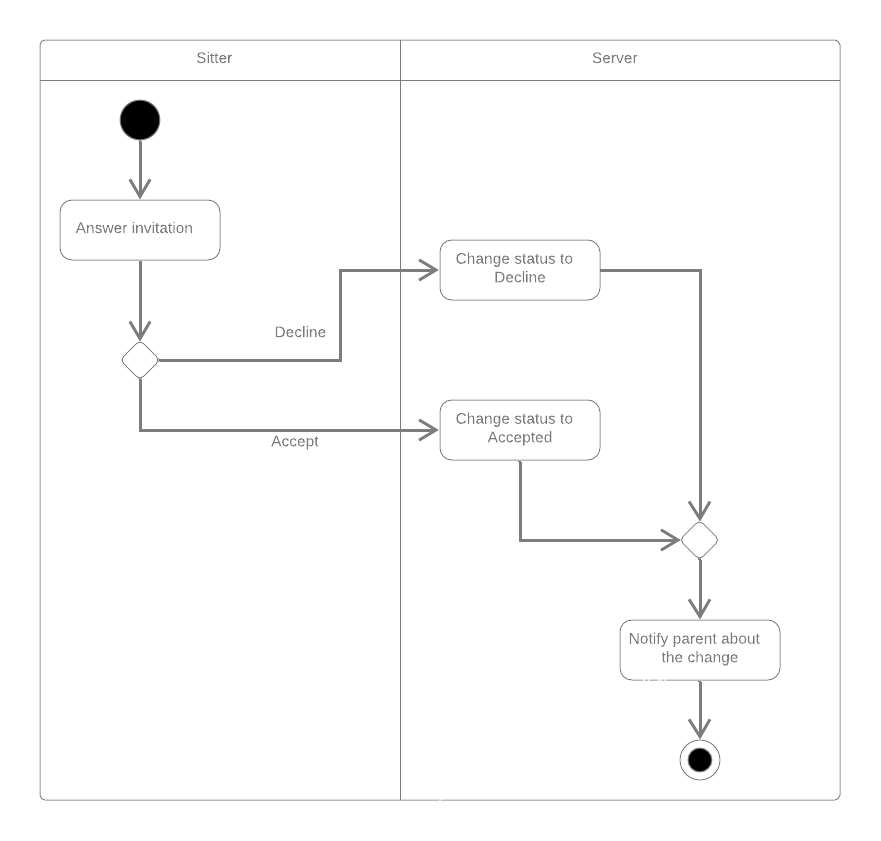


Figure 38 - <Babysitter> Answer invitation

#### <Parent> Create babysitting request

Summary: This diagram shows the process of parent create a babysitting request

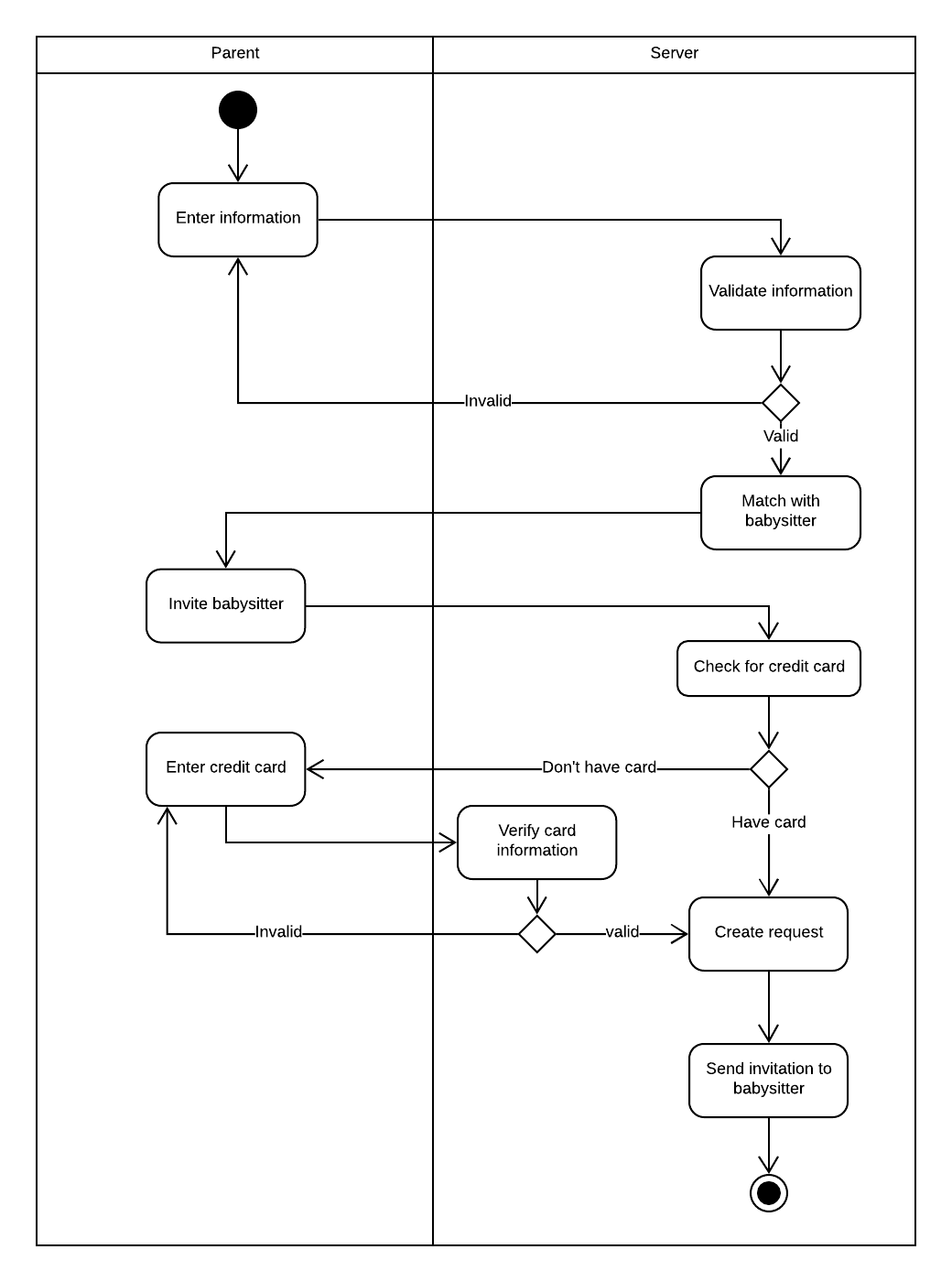


Figure 39: <Parent> Create babysitting request

#### <Parent> Accept babysitter

Summary: This diagram shows the process of parent accepting a babysitter for a request.

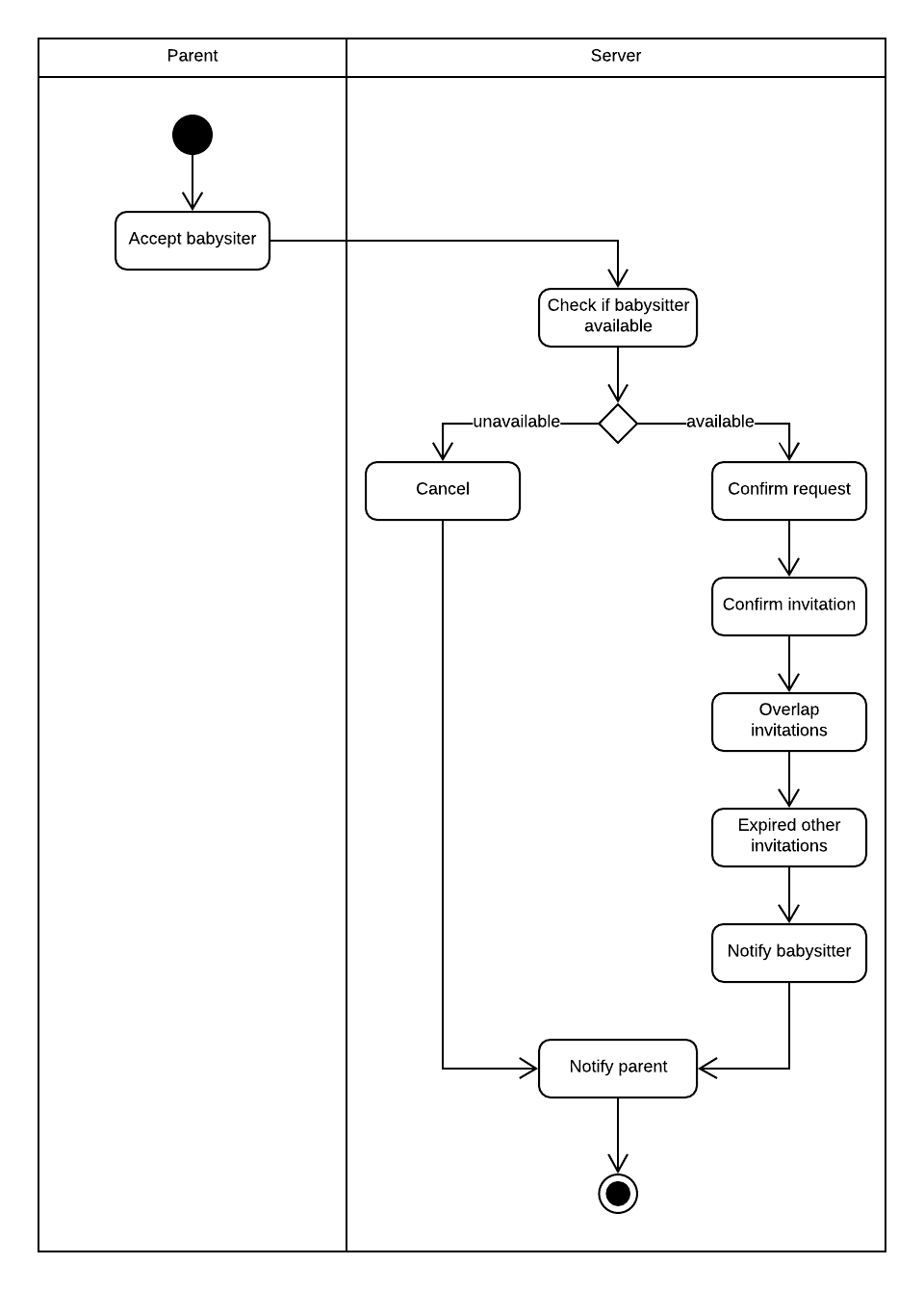


Figure 40 - <Parent> Accept babysitter

#### <Parent> Send feedback

Summary: This diagram shows the process of parent when send feedback

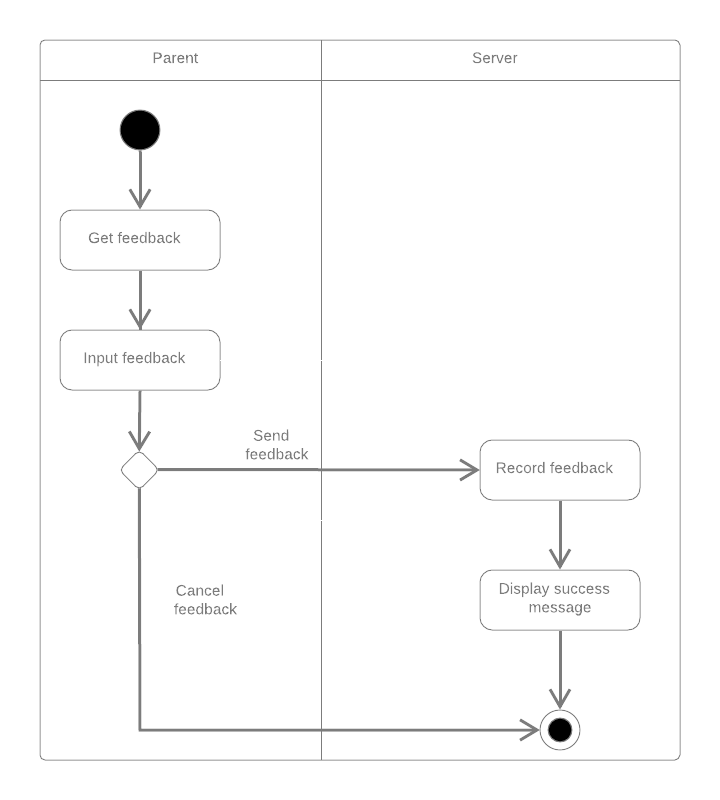


Figure 41 - <Parent> Send feedback

#### <Payment> Pay

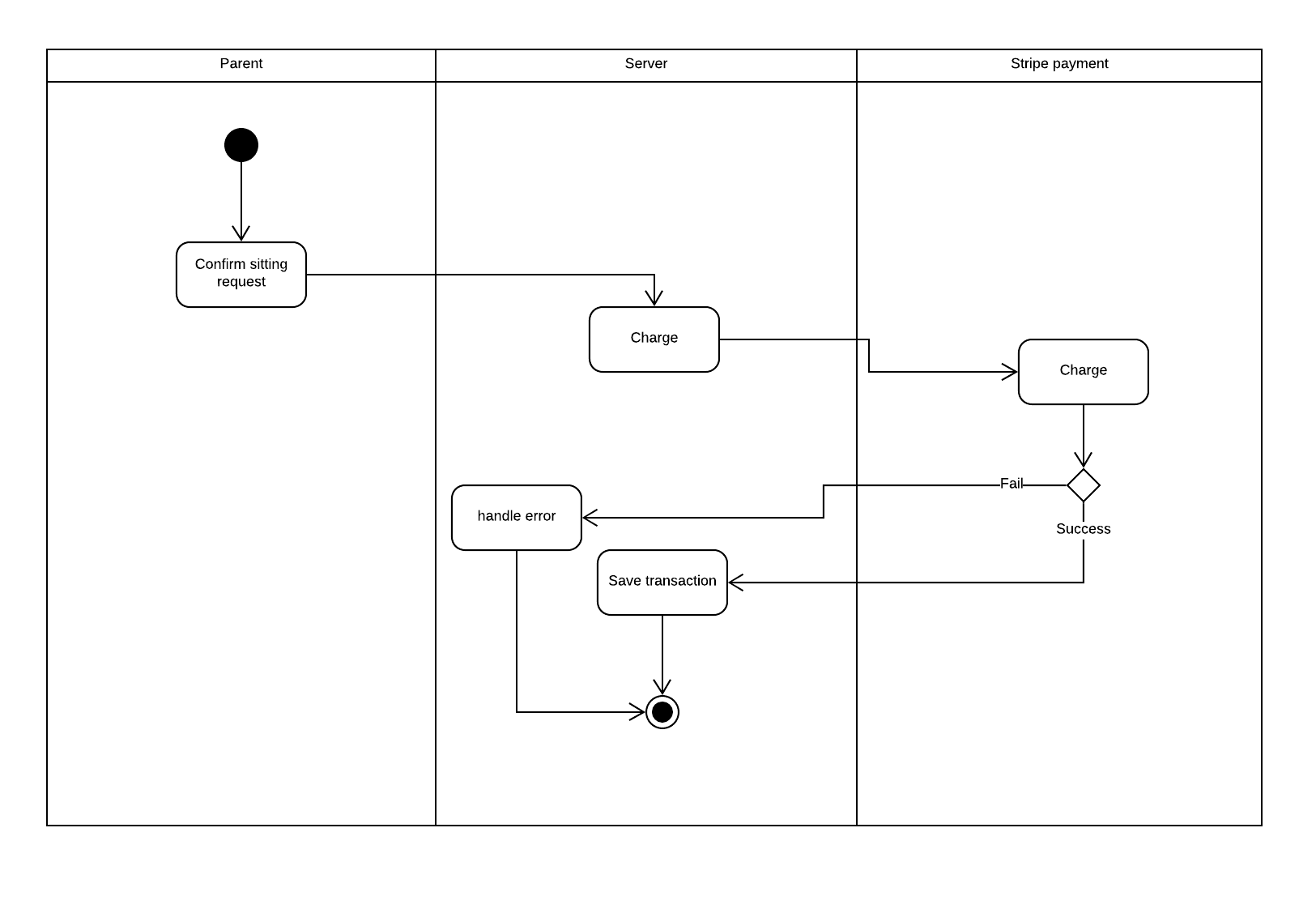
Summary: This diagram shows the process of paying a sitting request

Figure 42 - <Payment> Pay

#### <System Handler> Match parent and babysitter

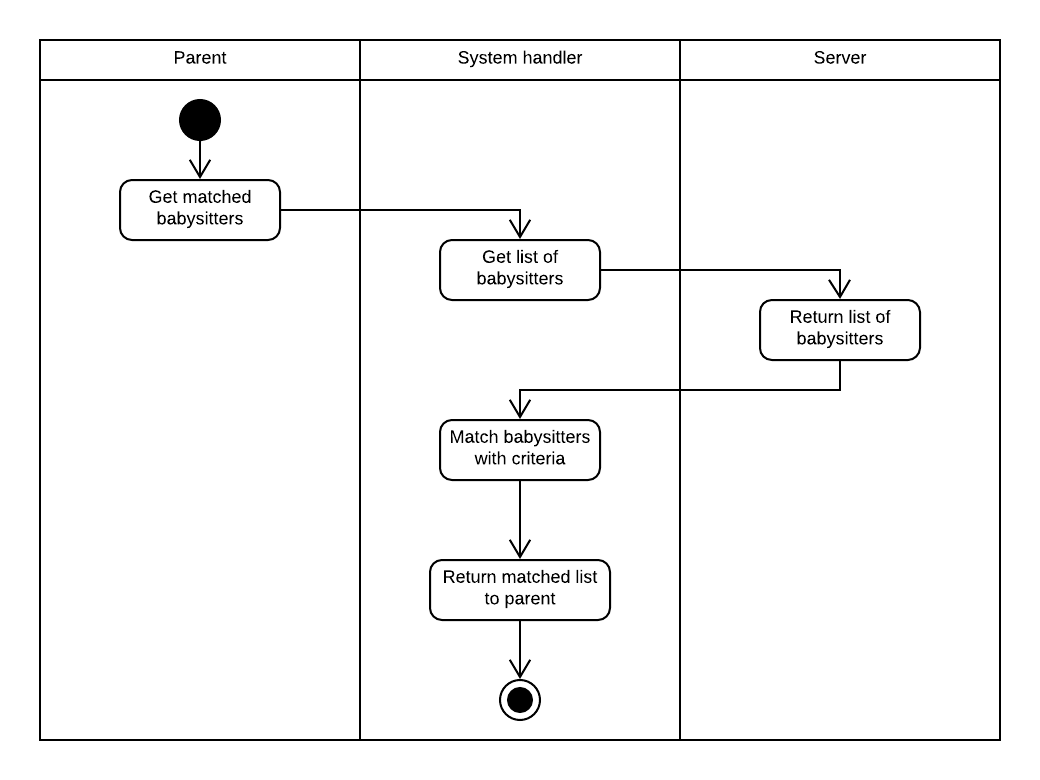
Summary: This diagram shows the process of matching a parent request and babysitters

Figure 43 - <System Handler> Match parent and babysitter

## Database Design

### Entity relationship diagram(ERD)

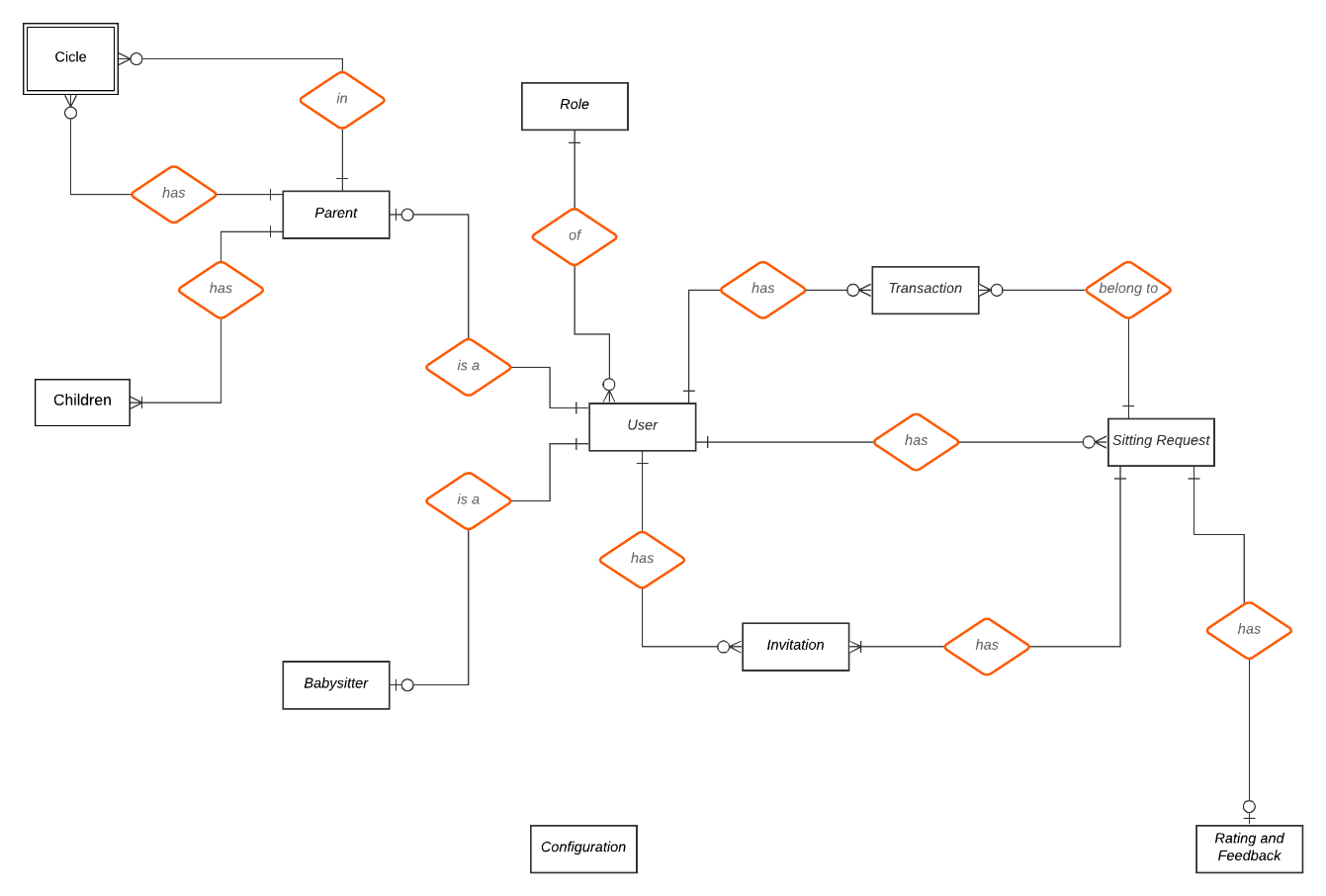


Figure 44 - Entity relationship diagram

|  |  |
| --- | --- |
| **Entity Data Dictionary** | |
| **Entity Name** | **Description** |
| User | Base entity of user in the system, contain user account information |
| Role | Role of the user |
| Parent | Represent role Parent in the system, contain information unique to Parent |
| Babysitter | Represent role Babysitter in the system, contain information unique to Babysitter |
| Children | Children of the Parent, contain information unique to Children |
| Circle | Represent trust circle of Parent, a Parent can have many other Parent in his/her trust circle and can be in other Parent trust circle as well |
| Sitting request | Represent the unit of work of the system |
| Rating Feedback | Feed back for a sitting request, given by the Parent |
| Transaction | Transaction between the Parent and the Babysitter though the system |
| Configuration | The configuration of the system |
| Credit card | Contain credit card information of user |

Table 36 - Data Dictionary

# **System Implementation & Test**

## Database relationship Diagram

### Physical Diagram

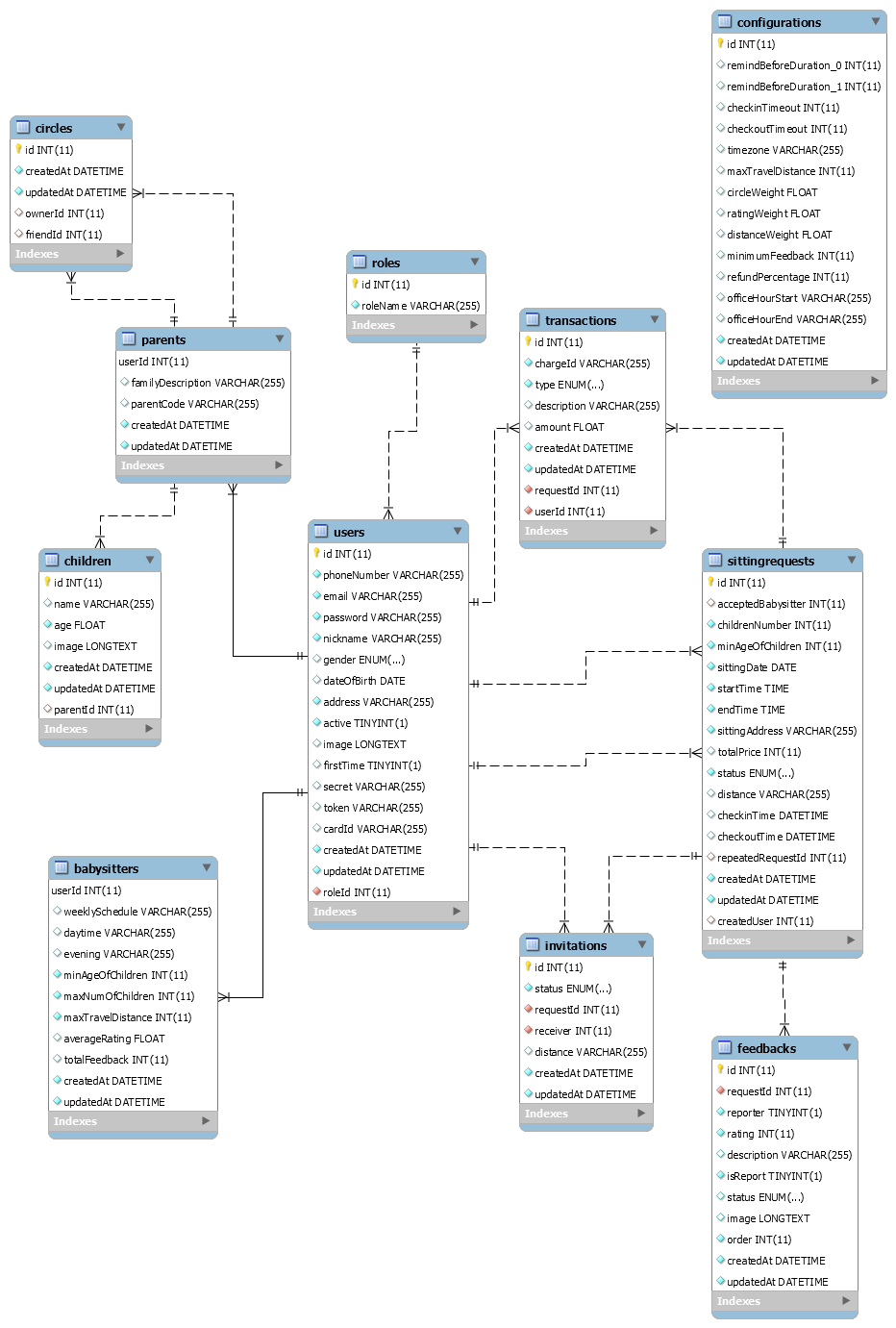


Figure 45 – Physical Diagram

|  |  |
| --- | --- |
| 1. **Entity Data Dictionary** | |
| **Entity Name** | **Description** |
| User | Base entity of user in the system, contain user account information |
| Role | Role of the user |
| Parent | Represent role Parent in the system, contain information unique to Parent |
| Babysitter | Represent role Babysitter in the system, contain information unique to Babysitter |
| Children | Children of the Parent, contain information unique to Children |
| Circle | Represent trust circle of Parent, a Parent can have many other Parent in his/her trust circle and can be in other Parent trust circle as well |
| Sitting request | Represent the unit of work of the system |
| Rating Feedback | Feed back for a sitting request, given by the Parent |
| Transaction | Transaction between the Parent and the Babysitter though the system |
| Configuration | The configuration of the system |
| Credit card | Contain credit card information of user |

Table 37 – Data dictionary

## Framework, technology

#### Node.js

In our system, most of process cases are **network calling** and wait then response. Node.js use an event-driven, non-blocking I/O model: **optimized for network intensive tasks**, which is verify different compared to Java, C#, Python which based on blocking I/O.

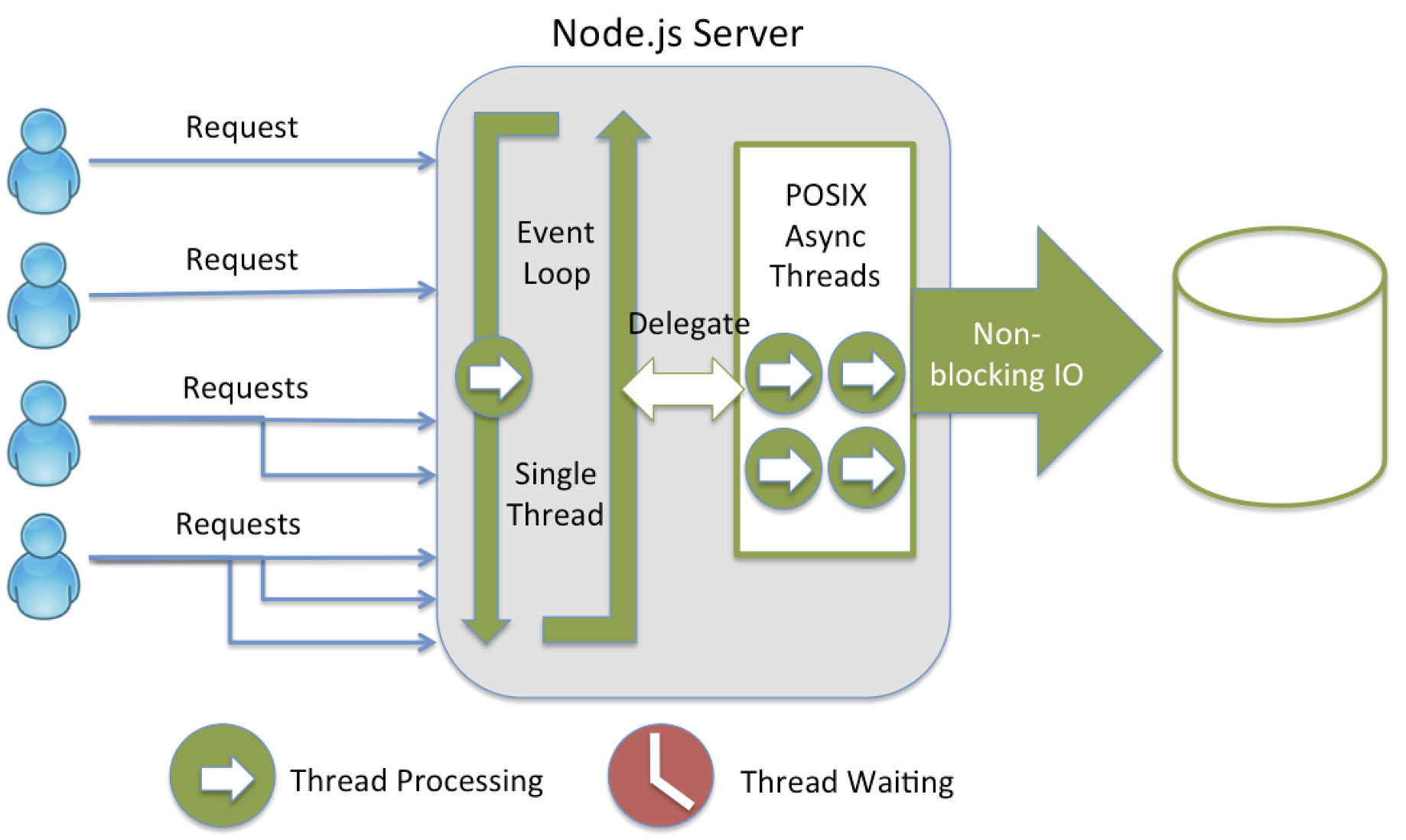


Figure 46 – NodeJS server(2)

#### Socket.io

To make the webapp real-time, we use socket.io library to implement the websocket protocol.

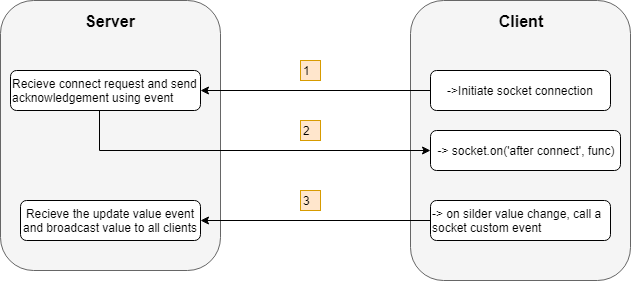


Figure 47 – Web. Socket communication with browser(4)

## Mobile Framework: React Native

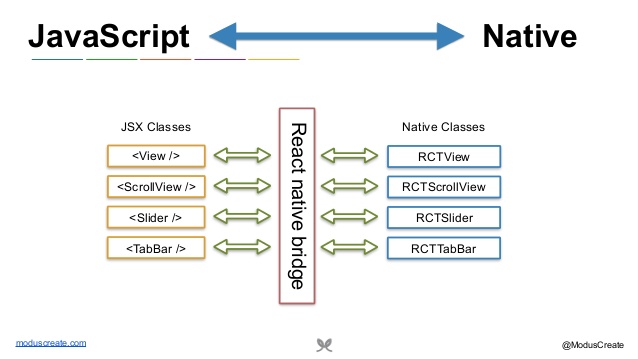


Figure 56 - React native bridge(3)

#### Time Efficiency

React Native has a feature called “hot reloading”. It allows the changes made in the code to immediately take effect into iOS and Android app variants. This reduces the waiting time for changes to show up to zero and saves further development time as well.

#### Performance

By using native controls and native modules, React Native improves on performance.

React Native interacts with the targeted (native) components for iOS or Android and renders code to native APIs directly and independently. In doing so, it uses a separate thread from UI, which results in increases the performance. The key factor here is React using the native APIs. Other options like using a WebView to render code may have other advantages, but compromise the performance.

#### Ability to Target Multiple OS with Less Native Code

The ability to create one app running for multiple OS is just great. No matter the simplicity because we might not get all the features or advanced functionality in apt packages for apps created with React Native, but nonetheless, we will have two applications with the time, effort and resources for creating one.

# **References**

**[1]** Dynamic Systems Development Method (DSDM) Phases, accessed 25 Sep 2019

<<https://electricalfundablog.com/agile-model-methodology/> >

**[2]** All About Node.Js You Wanted to Know, accessed 19 Sep 2019

<<https://codeburst.io/all-about-node-js-you-wanted-to-know-25f3374e0be7>>

< <https://nodejs.org/en/docs/>>

< https://expressjs.com/>

**[3]** React Native Documentation, accessed 21 Sep 2019

<<https://devdocs.io/react_native>>

< https://facebook.github.io/react-native/docs/getting-started>

**[4]** What Socket.IO is, accessed 15 Sep 2019

**<** https://socket.io/docs/ **>**