|  |  |
| --- | --- |
|  | **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** –

*This page is intentionally left blank*

# Table of Contents

[Table of Contents 3](#_Toc26986797)

[A. Introduction 11](#_Toc26986798)

[1. Project Information 11](#_Toc26986799)

[2. Introduction 11](#_Toc26986800)

[3. Current Situation 11](#_Toc26986801)

[4. Problem Definition 11](#_Toc26986802)

[5. Proposed Solution 12](#_Toc26986803)

[5.1. Feature functions 12](#_Toc26986804)

[5.2. Values and Challenges 13](#_Toc26986805)

[6. Functional Requirements 13](#_Toc26986806)

[6.1. Staff Component 13](#_Toc26986807)

[6.2. Parent Component 13](#_Toc26986808)

[6.3. Babysitter component 14](#_Toc26986809)

[6.4. Authentication component 14](#_Toc26986810)

[6.5. Recommendation component 14](#_Toc26986811)

[7. Role & Responsibility 14](#_Toc26986812)

[B. Software Project Management 14](#_Toc26986813)

[1. Problem Definition 14](#_Toc26986814)

[1.1. Name of this Capstone Project 14](#_Toc26986815)

[1.2. Problem Abstract 15](#_Toc26986816)

[1.3. Project Overview 15](#_Toc26986817)

[1.3.1. Current Situation 15](#_Toc26986818)

[1.3.2. The Proposed System 15](#_Toc26986819)

[1.3.3. Boundaries of the System 16](#_Toc26986820)

[1.3.4. Future Plans 17](#_Toc26986821)

[1.3.5. Development Environment 17](#_Toc26986822)

[1.3.5.1. Hardware requirements 17](#_Toc26986823)

[1.3.5.2. Software requirements 17](#_Toc26986824)

[2. Project Organization 18](#_Toc26986825)

[2.1. Software Process Model 18](#_Toc26986826)

[2.2. Roles and Responsibilities 20](#_Toc26986827)

[3. Project Management Plan 21](#_Toc26986828)

[3.1. Software development life cycle 21](#_Toc26986829)

[3.2. Phase Detail 22](#_Toc26986830)

[3.2.1. Phase 1: Requirements Definition 22](#_Toc26986831)

[3.2.2. Phase 2: System and Software Design 23](#_Toc26986832)

[3.2.3. Phase 3: Implementation and Unit Testing 23](#_Toc26986833)

[3.2.4. Phase 4: Integration and System Testing 24](#_Toc26986834)

[3.2.5. Phase 5: Operation and Maintenance 24](#_Toc26986835)

[3.3. Task sheet 25](#_Toc26986836)

[3.4. All Meeting Minutes 25](#_Toc26986837)

[4. Coding Convention 25](#_Toc26986838)

[C. Software Requirement Specification 26](#_Toc26986839)

[1. User Requirement Specification 26](#_Toc26986840)

[1.1. Guest requirement 26](#_Toc26986841)

[1.2. Parent requirement 26](#_Toc26986842)

[1.3. Babysitter requirement 26](#_Toc26986843)

[1.4. Staff requirement 26](#_Toc26986844)

[1.5. System handler requirement 27](#_Toc26986845)

[1.6. Authorized User 27](#_Toc26986846)

[1.7. Payment 27](#_Toc26986847)

[2. System Requirement Specification 27](#_Toc26986848)

[2.1. External Interface Requirement 27](#_Toc26986849)

[2.1.1. User Interface 27](#_Toc26986850)

[2.1.2. Hardware Interface 27](#_Toc26986851)

[2.1.3. Software Interface 27](#_Toc26986852)

[2.1.4. Communication Protocol 28](#_Toc26986853)

[2.2. System Overview Use Case 28](#_Toc26986854)

[2.3. List of Use Case 29](#_Toc26986855)

[2.3.1. <Guest> Login 29](#_Toc26986862)

[2.3.2. <Babysitter> Overview Use Case 31](#_Toc26986863)

[2.3.2.1. <Babysitter> Get profile 32](#_Toc26986864)

[2.3.2.2. <Babysitter> Edit sitting preferences 34](#_Toc26986865)

[2.3.2.3. <Babysitter> Answer invitation 36](#_Toc26986866)

[2.3.2.4. <Babysitter> Get invitation 38](#_Toc26986867)

[2.3.2.5. <Babysitter> Get invitation list 40](#_Toc26986868)

[2.3.2.6. <Babysitter> Get sitting history 42](#_Toc26986869)

[2.3.3. <Parent> Overview Use Case 44](#_Toc26986870)

[2.3.3.1. <Parent> Create babysitting request 45](#_Toc26986871)

[2.3.3.2. <Parent> Get babysitting request list 49](#_Toc26986872)

[2.3.3.3. <Parent> Get babysitting request 51](#_Toc26986873)

[2.3.3.4. <Parent> Accept babysitter 53](#_Toc26986874)

[2.3.3.5. <Parent> Add people to trust circle 56](#_Toc26986875)

[2.3.3.6. <Parent> Get request history 58](#_Toc26986876)

[2.3.3.7. <Parent> Get profile 60](#_Toc26986877)

[2.3.3.8. <Parent> Send feedback 62](#_Toc26986878)

[2.3.4. <Staff> Overview Use Case 64](#_Toc26986879)

[2.3.4.1. <Staff> Create user account 64](#_Toc26986880)

[2.3.4.2. <Staff> Update user information 67](#_Toc26986881)

[2.3.4.3. <Staff> Ban user 69](#_Toc26986882)

[2.3.4.4. <Staff> Edit configuration 71](#_Toc26986883)

[2.3.5. <Payment> Pay 74](#_Toc26986884)

[2.3.5.1. <Payment> Refund 75](#_Toc26986885)

[2.3.6. <System Handler> Overview Use Case 78](#_Toc26986886)

[2.3.6.1. <System Handler > Generate identification tokens 79](#_Toc26986887)

[2.3.6.2. <System Handler> Save transaction record 81](#_Toc26986888)

[2.3.6.3. <System Handler> Matching parent and babysitter 83](#_Toc26986889)

[2.3.6.4. <System Handler> Recommend babysitter to parent 86](#_Toc26986890)

[3. Software System Attribute 88](#_Toc26986891)

[3.1. Usability 88](#_Toc26986892)

[3.2. Reliability 88](#_Toc26986893)

[3.3. Availability 88](#_Toc26986894)

[3.4. Security 88](#_Toc26986895)

[3.5. Maintainability 88](#_Toc26986896)

[3.6. Portability 88](#_Toc26986897)

[3.7. Performance 89](#_Toc26986898)

[4. Conceptual Diagram 89](#_Toc26986899)

[D. Software Design Description 90](#_Toc26986900)

[1. Design Overview 90](#_Toc26986901)

[2. System Architectural Design 90](#_Toc26986902)

[3. Component Diagram 91](#_Toc26986903)

[4. Detail Description 93](#_Toc26986904)

[4.1. Class Diagram 93](#_Toc26986905)

[4.2. Class Diagram Explanation 94](#_Toc26986906)

[4.2.1. User 94](#_Toc26986907)

[4.2.2. Parent 95](#_Toc26986908)

[4.2.3. Children 95](#_Toc26986909)

[4.2.4. Circle 96](#_Toc26986910)

[4.2.5. Babysitter 96](#_Toc26986911)

[4.2.6. Role 97](#_Toc26986912)

[4.2.7. SittingRequest 97](#_Toc26986913)

[4.2.8. Feedback 97](#_Toc26986914)

[4.2.9. Invitation 98](#_Toc26986915)

[4.2.10. Configuration 98](#_Toc26986916)

[4.2.11. Transaction 99](#_Toc26986917)

[4.3. Interaction Diagram 99](#_Toc26986918)

[4.3.1. <Guest> Login 100](#_Toc26986919)

[4.3.2. <Babysitter> Answer invitation 101](#_Toc26986920)

[4.3.3. <Parent> Create babysitting request 102](#_Toc26986921)

[4.3.4. <Parent> Accept babysitter 103](#_Toc26986922)

[4.3.5. <Parent> Send feedback 104](#_Toc26986923)

[4.3.6. <Payment> Pay 104](#_Toc26986924)

[4.3.7. <System Handler> Match parent and babysitter 106](#_Toc26986925)

[5. Interface 107](#_Toc26986926)

[5.1. Mobile application design 107](#_Toc26986927)

[5.1.1. Parent and Sitter login screen 107](#_Toc26986928)

[5.1.1.1. Input OTP popup 109](#_Toc26986929)

[5.1.2. Parent Interface design 111](#_Toc26986930)

[5.1.2.1. Parent Home screen 111](#_Toc26986931)

[5.1.2.2. Create request screen 112](#_Toc26986932)

[5.1.2.3. Recommend screen 114](#_Toc26986933)

[5.1.2.4. Bank card popup 115](#_Toc26986934)

[5.1.2.5. Profile Sitter Recommend Detail 117](#_Toc26986935)

[5.1.2.6. Request Detail 119](#_Toc26986936)

[5.1.3. Sitter interface design 122](#_Toc26986937)

[5.1.3.1. Sitter home screen 123](#_Toc26986938)

[5.1.3.2. Invitation Detail 124](#_Toc26986939)

[5.1.3.3. QR code Babysitter 125](#_Toc26986940)

[6. Database Design 126](#_Toc26986941)

[6.1. Entity relationship diagram (ERD) 126](#_Toc26986942)

[6.2. Data Dictionary 126](#_Toc26986943)

[E. System implementation and Testing 131](#_Toc26986944)

[1. Introduction 131](#_Toc26986945)

[1.1. Overview 131](#_Toc26986946)

[1.2. Test Approach 131](#_Toc26986947)

[2. Database relationship Diagram 132](#_Toc26986948)

[2.1. Physical Diagram 132](#_Toc26986949)

[2.2. Data Dictionary 134](#_Toc26986950)

[3. Performance Measures 137](#_Toc26986951)

[3.1. Web application page load speed 137](#_Toc26986952)

[3.1.1. Definition 137](#_Toc26986953)

[3.1.2. Test environment 137](#_Toc26986954)

[3.1.3. Test cases 137](#_Toc26986955)

[3.1.4. Test result 138](#_Toc26986956)

[3.2. Mobile application API load speed 138](#_Toc26986957)

[3.2.1. Definition 138](#_Toc26986958)

[3.2.2. Test environment 138](#_Toc26986959)

[3.2.3. Test cases 139](#_Toc26986960)

[3.2.4. Test result 139](#_Toc26986961)

[4. Test Plan 140](#_Toc26986962)

[4.1. Features to be tested 140](#_Toc26986963)

[4.2. Features not to be tested 140](#_Toc26986964)

[5. System testing test case 142](#_Toc26986965)

[5.1. Communication Diagrams 142](#_Toc26986966)

[5.2. Test case 142](#_Toc26986967)

[5.2.1. <Parent> Create request 142](#_Toc26986968)

[5.2.2. <Parent> Payment 144](#_Toc26986969)

[5.2.3. <Sitter> Answer invitation 144](#_Toc26986970)

[5.2.4. <Parent > Accept babsitter 144](#_Toc26986971)

[5.2.5. <Parent> Recommend babysitter 144](#_Toc26986972)

[5.3. Test case results statistics 145](#_Toc26986973)

[5.4. Framework, technology 145](#_Toc26986974)

[5.4.1. Node.js 145](#_Toc26986975)

[5.4.2. Socket.io 146](#_Toc26986976)

[5.5. Mobile Framework: React Native 147](#_Toc26986977)

[5.5.1. Time Efficiency 147](#_Toc26986978)

[5.5.2. Performance 147](#_Toc26986979)

[5.5.3. Ability to Target Multiple OS with Less Native Code 147](#_Toc26986980)

[F. Software User’s Manual 147](#_Toc26986981)

[1. Installation Guide 147](#_Toc26986982)

[1.1. Setting up environment at server side 147](#_Toc26986983)

[1.1.1. Hardware requirements 147](#_Toc26986984)

[1.1.2. Software requirements 147](#_Toc26986985)

[1.2. Deployment at Production server 148](#_Toc26986986)

[1.3. Setting up the environment at client side 148](#_Toc26986987)

[Appendix 149](#_Toc26986988)

List of Figures

[Figure 1 - Atern Development Model - DSDM 19](#_Toc26986989)

[Figure 2 - Software development life cycle 21](#_Toc26986990)

[Figure 3 - Overview Use Case 28](#_Toc26986991)

[Figure 4 - <Guest> Login 29](#_Toc26986992)

[Figure 5 - <Babysitter> Overview Use Case 31](#_Toc26986993)

[Figure 6 - <Babysitter> Get profile 32](#_Toc26986994)

[Figure 7 - <Babysitter> Edit sitting preferences 34](#_Toc26986995)

[Figure 8 - <Babysitter> Answer invitation 36](#_Toc26986996)

[Figure 9 - <Babysitter>Get invitation 38](#_Toc26986997)

[Figure 10 - <Babysitter> Get invitation list 40](#_Toc26986998)

[Figure 11 - <Babysitter> Get sitting history 42](#_Toc26986999)

[Figure 12 - <Parent> Overview Use Case 44](#_Toc26987000)

[Figure 13 - <Parent> Create babysitting request 45](#_Toc26987001)

[Figure 14 - <Parent> Get babysitting request list 49](#_Toc26987002)

[Figure 15 - <Parent> Get babysitting request 51](#_Toc26987003)

[Figure 16 - <Parent> Accept babysitter 53](#_Toc26987004)

[Figure 17 - <Parent> Add people to trust circle 56](#_Toc26987005)

[Figure 18 - <Parent> Get request history 58](#_Toc26987006)

[Figure 19 - <Parent> Get profile 60](#_Toc26987007)

[Figure 20 - <Parent> Send feedback 62](#_Toc26987008)

[Figure 21 - <Staff> Overview Use Case 64](#_Toc26987009)

[Figure 22 - <Staff> Create user account 64](#_Toc26987010)

[Figure 23 - <Staff> Update user information 67](#_Toc26987011)

[Figure 24 - <Staff> Ban user 70](#_Toc26987012)

[Figure 25 - <Staff> Edit configuration 71](#_Toc26987013)

[Figure 26 - <Payment> Pay 74](#_Toc26987014)

[Figure 27 - <Payment> Refund 76](#_Toc26987015)

[Figure 28 - <System Handler> Overview Use Case 78](#_Toc26987016)

[Figure 29 - <System Handler > Generate identification tokens 79](#_Toc26987017)

[Figure 30 - <System Handler> Save transaction record 81](#_Toc26987018)

[Figure 31 - <System Handler> Matching parent and babysitter 83](#_Toc26987019)

[Figure 32 - <System Handler> Recommend babysitter to parent 86](#_Toc26987020)

[Figure 33 - Conceptual Diagram 89](#_Toc26987021)

[Figure 34 - System architecture design 90](#_Toc26987022)

[Figure 35 - Front-end component diagram 91](#_Toc26987023)

[Figure 36 - Back-end component diagram 92](#_Toc26987024)

[Figure 37 - Class Diagram 93](#_Toc26987025)

[Figure 38 - <Guest> Login 100](#_Toc26987026)

[Figure 39 - <Babysitter> Answer invitation 101](#_Toc26987027)

[Figure 40: <Parent> Create babysitting request 102](#_Toc26987028)

[Figure 41 - <Parent> Accept babysitter 103](#_Toc26987029)

[Figure 42 - <Parent> Send feedback 104](#_Toc26987030)

[Figure 43 - <Payment> Pay 105](#_Toc26987031)

[Figure 44 - <System Handler> Match parent and babysitter 106](#_Toc26987032)

[Figure 45 – Entity relationship diagram (ERD) 126](#_Toc26987033)

[Figure 46 - Physical Diagram 134](#_Toc26987034)

[Figure 47 - <Unauthorized User> authentication flow 142](#_Toc26987035)

[Figure 48 - <Parent> Create request flow 142](#_Toc26987036)

[Figure 49 - <Sitter> Answer invitation flow 142](#_Toc26987037)

[Figure 50 – NodeJS server 146](#_Toc26987038)

[Figure 51 – Web. Socket communication with browser 146](#_Toc26987039)

List of Tables

[Table 1 - Definitions, Acronyms, and Abbreviations 10](#_Toc26987040)

[Table 2 - Roles and Responsibilities 14](#_Toc26987041)

[Table 3 - Server 17](#_Toc26987042)

[Table 4 - Computer 17](#_Toc26987043)

[Table 5 - Web 17](#_Toc26987044)

[Table 6 - Mobile application 18](#_Toc26987045)

[Table 7 - Key responsibilities 19](#_Toc26987046)

[Table 8 - Role and Responsibilities 21](#_Toc26987047)

[Table 9 – Phase overview 22](#_Toc26987048)

[Table 10 - Phase 1: Requirement Definition 23](#_Toc26987049)

[Table 11 – Phase 2: System and Software Design 23](#_Toc26987050)

[Table 12 – Phase 3: Implementation and Unit Testing 24](#_Toc26987051)

[Table 13 – Phase 4: Integration and System Testing 24](#_Toc26987052)

[Table 14 – Phase 5: Operation and Maintenance 24](#_Toc26987053)

[Table 15 - Use case LG.01 - <Guest> Login 31](#_Toc26987054)

[Table 16 - Use case BS.01 - <Babysitter> Get profile 33](#_Toc26987055)

[Table 17 - Use case BS.02 - <Babysitter> Edit sitting preferences 35](#_Toc26987056)

[Table 18 - Use case BS.03 - <Babysitter> Answer invitation 38](#_Toc26987057)

[Table 19 - Use case BS.04 - <Babysitter> Get invitation 40](#_Toc26987058)

[Table 20 - Use case BS.05 - <Babysitter> Get invitation list 42](#_Toc26987059)

[Table 21 - Use case BS.06 - <Babysitter> Get sitting history 43](#_Toc26987060)

[Table 22 - Use case PR.01 - <Parent> Create babysitting request 48](#_Toc26987061)

[Table 23 - Use case PR.02 - <Parent> Get babysitting request list 51](#_Toc26987062)

[Table 24 - Use case PR.03 - <Parent> Get babysitting request 53](#_Toc26987063)

[Table 25 - Use case PR.04 - <Parent> Accept babysitter 55](#_Toc26987064)

[Table 26 - Use case PR.05 - <Parent> Add people to trust circle 57](#_Toc26987065)

[Table 27 - Use case PR.06 - <Parent> Get request history 59](#_Toc26987066)

[Table 28 - Use case PR.07 - <Parent> Get profile 61](#_Toc26987067)

[Table 29 - Use case PR.08 - <Parent> Send feedback 64](#_Toc26987068)

[Table 30 - Use case ST.01 - <Staff> Create user account 67](#_Toc26987069)

[Table 31 - Use case ST.02 - <Staff> Update user information 69](#_Toc26987070)

[Table 32 - Use case ST.03 - <Staff> Ban user 71](#_Toc26987071)

[Table 33 - Use case ST.04 - <Staff> Edit configuration 73](#_Toc26987072)

[Table 34 - Use case PM.01 - <Payment> Pay 75](#_Toc26987073)

[Table 35 - Use case PM.02 - <Payment> Refund 77](#_Toc26987074)

[Table 36 - Use case SH.01 - <System Handler > Generate identification tokens 80](#_Toc26987075)

[Table 37 - Use case SH.02 - <System Handler> Save transaction record 82](#_Toc26987076)

[Table 38 - Use case SH.03 - <System Handler> Matching parent and babysitter 85](#_Toc26987077)

[Table 39 - Use case SH.04 - <System Handler> Recommend babysitter to parent 88](#_Toc26987078)

[Table 40 - Data dictionary 90](#_Toc26987079)

[Table 41 - Front-end component diagram dictionary 91](#_Toc26987080)

[Table 42 - Back-end component diagram dictionary 92](#_Toc26987081)

[Table 43 - Class Dictionary 94](#_Toc26987082)

[Table 44 - User Attribute 94](#_Toc26987083)

[Table 45 - User Method 95](#_Toc26987084)

[Table 46 -Parent Attribute 95](#_Toc26987085)

[Table 47 - Parent Method 95](#_Toc26987086)

[Table 48 - Children Attribute 95](#_Toc26987087)

[Table 49 - Children Method 95](#_Toc26987088)

[Table 50 -Circle Attribute 96](#_Toc26987089)

[Table 51 -Circle Method 96](#_Toc26987090)

[Table 52 -Babysitter Attribute 96](#_Toc26987091)

[Table 53 - Babysitter Method 96](#_Toc26987092)

[Table 54 -Role Attribute 97](#_Toc26987093)

[Table 55 -Role Method 97](#_Toc26987094)

[Table 56 -SittingRequest Attribute 97](#_Toc26987095)

[Table 57 – SittingRequest Method 97](#_Toc26987096)

[Table 58 -Feedback Attribute 98](#_Toc26987097)

[Table 59 -Feedback Method 98](#_Toc26987098)

[Table 60 -Invitation Attribute 98](#_Toc26987099)

[Table 61 - Invitation Method 98](#_Toc26987100)

[Table 62 -Configuration Attribute 99](#_Toc26987101)

[Table 63 - Configuration Method 99](#_Toc26987102)

[Table 64 -Transaction Attribute 99](#_Toc26987103)

[Table 65 -Transaction Method 99](#_Toc26987104)

[Table 66 - Data Dictionary 127](#_Toc26987105)

[Table 67- User 127](#_Toc26987106)

[Table 68 - Role 127](#_Toc26987107)

[Table 69 - Parent 127](#_Toc26987108)

[Table 70 - Babysitter 128](#_Toc26987109)

[Table 71 - Children 128](#_Toc26987110)

[Table 72 - Circle 128](#_Toc26987111)

[Table 73 - Sitting Request 128](#_Toc26987112)

[Table 74 - Rating and Feedback 129](#_Toc26987113)

[Table 75 - Transaction 129](#_Toc26987114)

[Table 76 - Configuration 130](#_Toc26987115)

[Table 77 - Entity Data Dictionary 134](#_Toc26987116)

[Table 78 - User 134](#_Toc26987117)

[Table 79 - Role 134](#_Toc26987118)

[Table 80 - Parent 135](#_Toc26987119)

[Table 81 - Parent 135](#_Toc26987120)

[Table 82 - Children 135](#_Toc26987121)

[Table 83 - Circle 135](#_Toc26987122)

[Table 84 - Sitting Request 136](#_Toc26987123)

[Table 85 - Rating and Feedback 136](#_Toc26987124)

[Table 86 - Transaction 136](#_Toc26987125)

[Table 87 - Configuration 137](#_Toc26987126)

[Table 88 - Test pages web application 138](#_Toc26987127)

[Table 89 - Web application page load speed test result 138](#_Toc26987128)

[Table 90 - Mobile app API load speed client specification 139](#_Toc26987129)

[Table 91 - Mobile app API load speed test cases 139](#_Toc26987130)

[Table 92 - Mobile application API load speed test result 140](#_Toc26987131)

[Table 93 - Test case <Parent> Create request 143](#_Toc26987132)

[Table 94 - Test case <Parent> Payment 144](#_Toc26987133)

[Table 95 - Test case <Sitter> Accept invitation 144](#_Toc26987134)

[Table 96 - Test case <Parent> Accept babysitter 144](#_Toc26987135)

[Table 97 - <Parent> Recommend 145](#_Toc26987136)

[Table 98 - Test case results statistics 145](#_Toc26987137)

[Table 99 - Hardware requirements 147](#_Toc26987138)

[Table 100 - Software requirements 148](#_Toc26987139)

|  |  |
| --- | --- |
| **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

# Introduction

## 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.
* **Calculate babysitter point:** Automatically calculates points when the babysitter completes a task assigned.

### 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

## Role & Responsibility

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **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 2 - Roles and Responsibilities

# Software Project Management

## 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

#### Development Environment

##### Hardware requirements

**For Server**

|  |  |  |
| --- | --- | --- |
| **Windows** | **Minimum Requirement** | **Recommended** |
| **Internet Connection** | Cable, Wi-Fi (6 Mbps) | Cable, Wi-Fi (16 Mbps) |
| **Operating System** | Ubuntu 18.04 LTS or above | Ubuntu 18.04 LTS or above |
| **Computer Processor** | Core i7 (5th) | Core i7 (7th) |
| **Computer Memory** | 8GB RAM | 16GB RAM |

Table 3 - Server

**For Computer**

|  |  |  |
| --- | --- | --- |
| **Windows** | **Minimum Requirement** | **Recommended** |
| **Internet Connection** | Cable, Wi-Fi (6 Mbps) | Cable, Wi-Fi (16 Mbps) |
| **Operating System** | Windows 7 or above | Windows 7 or above |
| **Computer Processor** | Core i3 (5th). | Core i5 (5th) |
| **Computer Memory** | 4GB RAM | 8GB RAM |

Table 4 - Computer

##### Software requirements

**For Web**

|  |  |  |
| --- | --- | --- |
| **Software** | **Name / Version** | **Description** |
| **Operating system** | MacOS, Windows 7 or above, ... | Support for multiple Systems. |
| **Environment** | Nodejs | Specification for backend development |
| **Modeling tool** | Lucid Chart, Adobe Xd | Used to implement diagrams and prototypes |
| **IDE** | Visual Studio Code | Visual Studio IDE |
| **DBMS** | MySQL Database | MySQL database |
| **Source control** | GitHub | Github.com |
| **Web browser** | Chrome, Firefox, Safari, Edge | Support for multiple platform. |

Table 5 - Web

**For Mobile Application**

|  |  |  |
| --- | --- | --- |
| **Software** | **Name / Version** | **Description** |
| **Operating system** | Android OS 6.0 or above, IOS 10 or above. | Support for both Android and IOS. |
| **Environment** | Android or IOS environment. | Mobile device using Android or IOS system. |
| **Modeling tool** | Android, IOS or Chrome browser. | Android or IOS device. |
| **IDE** | Visual Code | Visual Code IDE. |
| **DBMS** | Local Storage | Android local storage. |
| **Source control** | GitHub | github.com |

Table 6 - Mobile application

## 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:

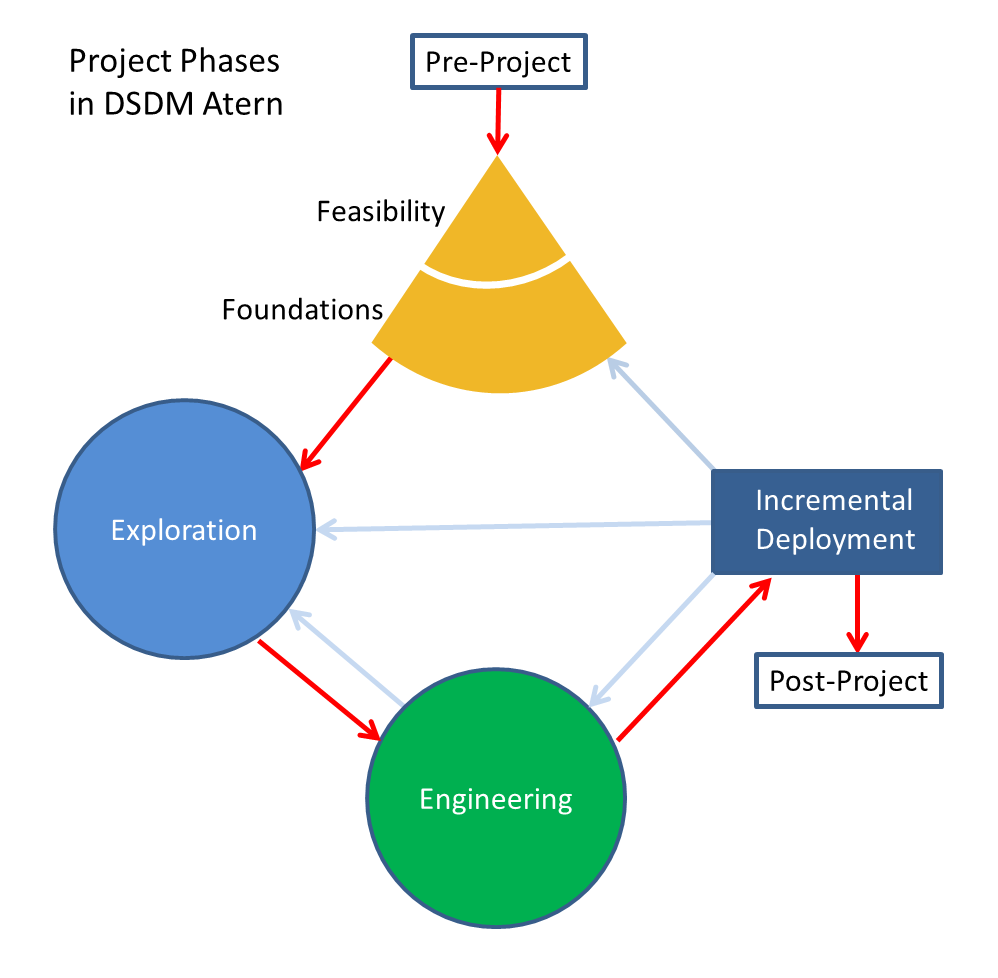


Figure 1 - Atern Development Model - DSDM

|  |  |
| --- | --- |
| 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 7 - Key responsibilities

### 2.2. Roles and Responsibilities

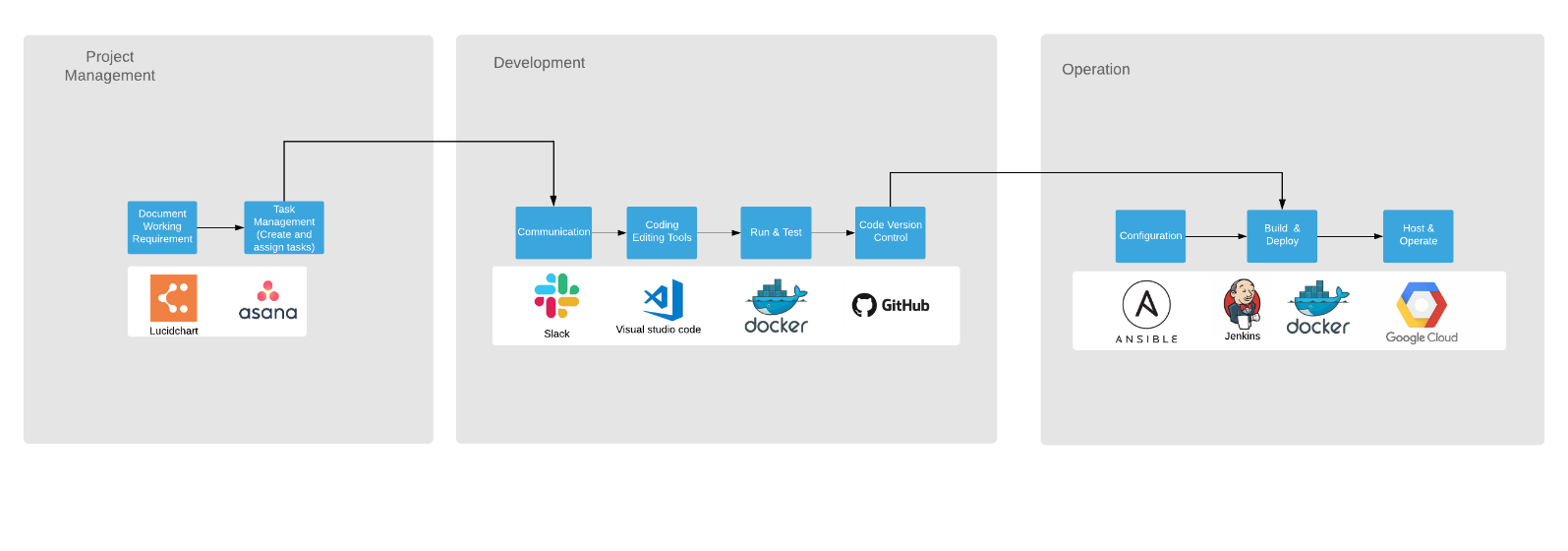
|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Full name** | **Role in Group** | **Responsibilities** |
| 1 | Kiều Trọng Khánh | Project Owner | • Specify user requirement  • Control the development process  • Support in technique and business logic |
| 2 | Phùng Thiên Phúc – SE62639 (Leader) | Team Leader, BA, Developer, Tester | • Managing process  • Designing database  • Clarifying requirements  • Give out technique and business analysis support  • Prepare documents  • GUI Design  • Create test plan  • Coding  • Testing |
| 3 | Phạm Hải Dương – SE62746 | Developer/Tester | • Designing database  • Clarifying requirements  • Prepare documents  • Coding  • Testing |
| 4 | Hồ Tấn Kỳ – SE61974 | Developer/Tester | • Clarifying requirements  • Prepare documents  • GUI Design  • Coding  • Testing |
| 5 | Hoàng Nhật Đông – SE62357 | Developer/Tester | • Designing database  • Prepare documents  • GUI Design  • Coding  • Testing |

Table 8 - Role and 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 9 – 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 10 - 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 11 – 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 12 – 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 13 – 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 14 – Phase 5: Operation and Maintenance

### Task sheet

Refer to “Task sheet” folder: https://bom.to/RQAoID

### All Meeting Minutes

Refer to “Meeting minutes” folder: *https://bom.to/RQAoID*

## 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

### External Interface Requirement

#### User Interface

* GUI should be simple, clear, intuitive, and reminiscent.
* The interface design is an iterate process includes: design, sketching, prototyping, user assessment.
* The User Interface uses language is Vietnamese.

The web administration user interface will be in English and contain the following components:

* Pages for staff to edit information and manage parent, sitter.
* Pages for staff to manage their information.
* Intuitive data tables for staff to manage positions, skills.
* Pages for administrator to manage staff.

#### Hardware Interface

* Desktop PC
* Mobile Device
* A browser which supports CGI, HTML & JavaScript.

#### Software Interface

* At least android 6.0.
* Web application: work with browsers Firefox (v52 or higher), Chromes (v28 or higher), Internet Explorer (v10 or above) or with any web browser that supports HTML5 & CSS3.

#### Communication Protocol

* Use HTTP protocol (version 1.1) for communication between web services and web application or mobile device application that consumes the services.

### System Overview Use Case

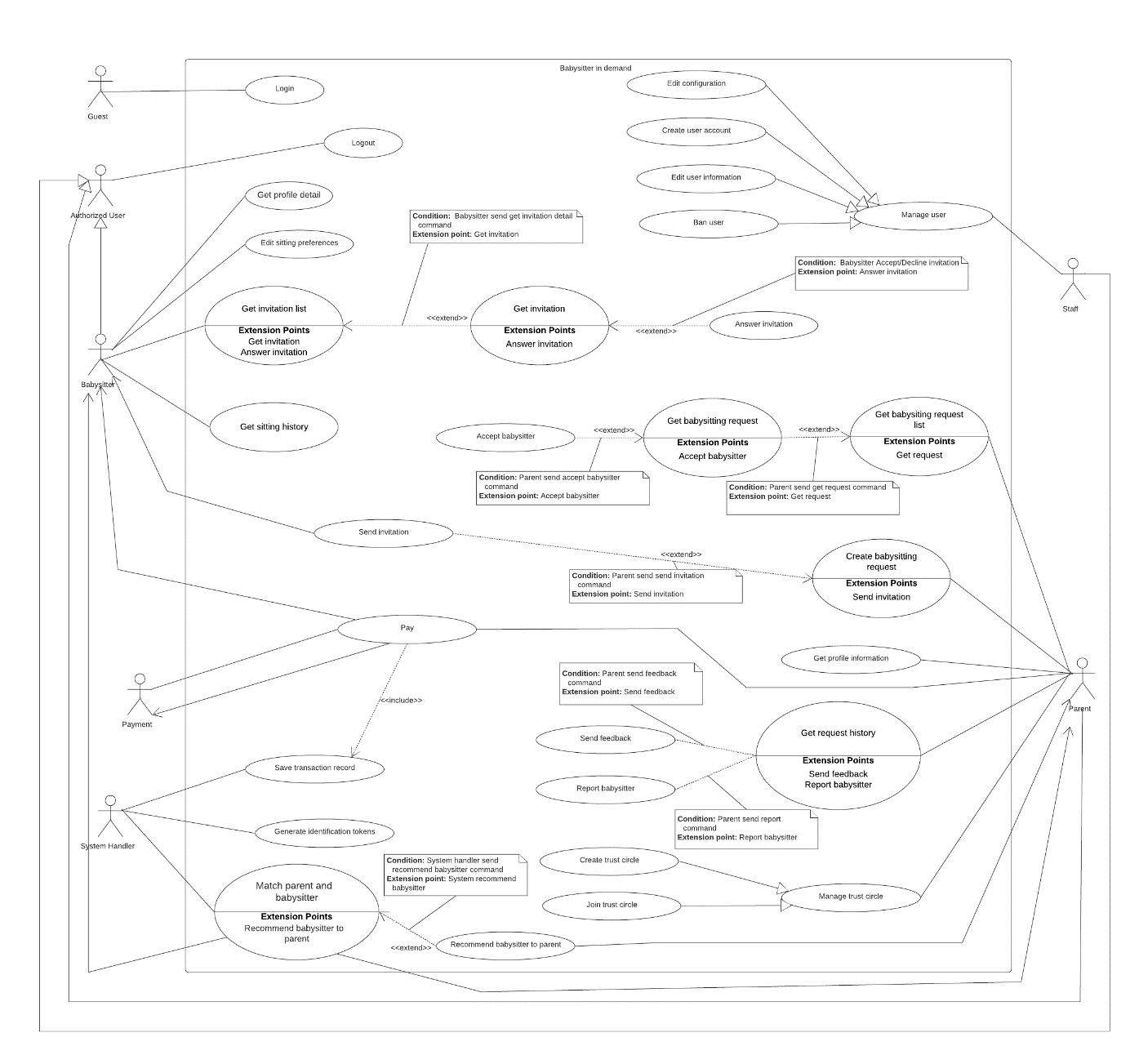


Figure 3 - Overview Use Case

### List of Use Case

#### <Guest> Login

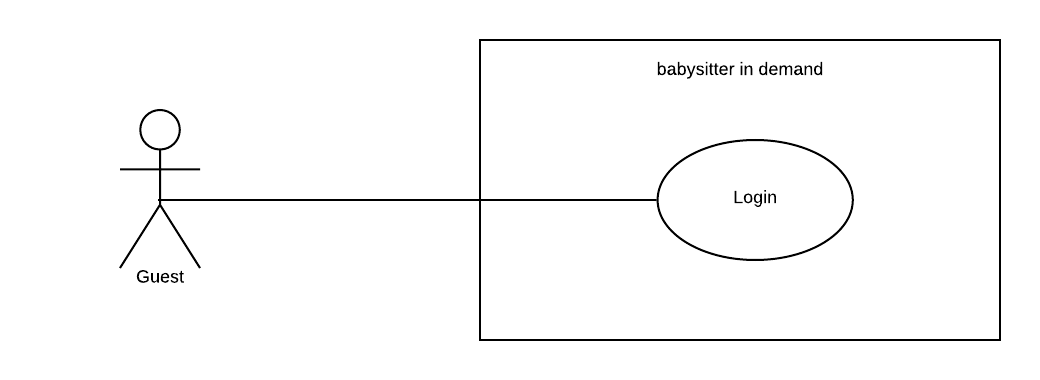


Figure 4 - <Guest> Login

|  |  |  |  |
| --- | --- | --- | --- |
| **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 15 - Use case LG.01 - <Guest> Login

#### <Babysitter> Overview Use Case

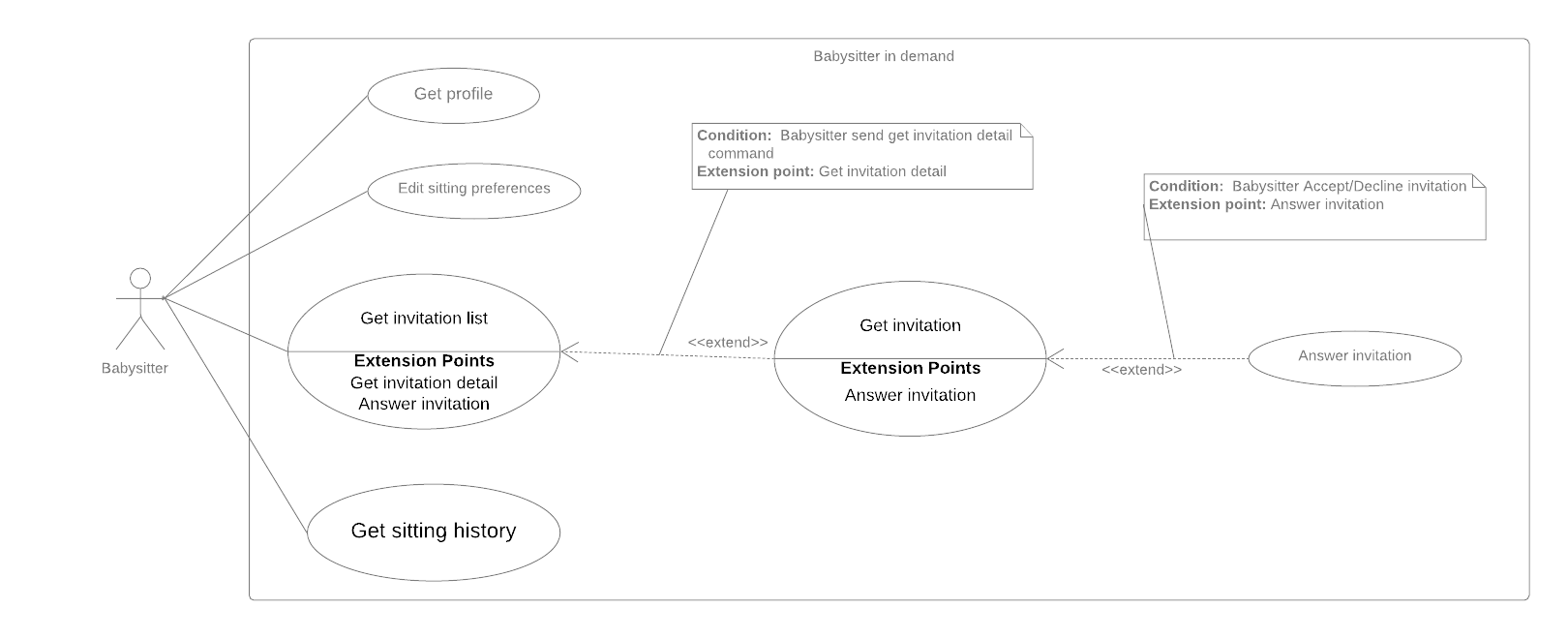


Figure 5 - <Babysitter> Overview Use Case

##### <**Babysitter> Get profile**

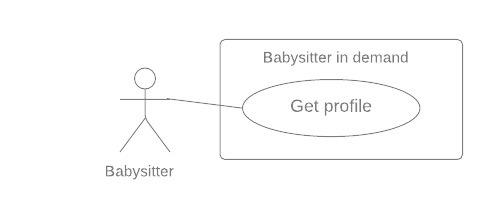


Figure 6 - <Babysitter> Get profile

|  |  |  |  |
| --- | --- | --- | --- |
| **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 16 - Use case BS.01 - <Babysitter> Get profile



##### <Babysitter> Edit sitting preferences

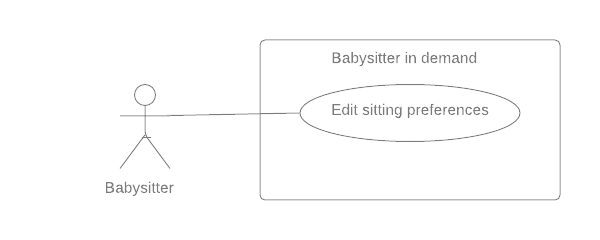


Figure 7 - <Babysitter> Edit sitting preferences

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **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 17 - Use case BS.02 - <Babysitter> Edit sitting preferences

##### <Babysitter> Answer invitation

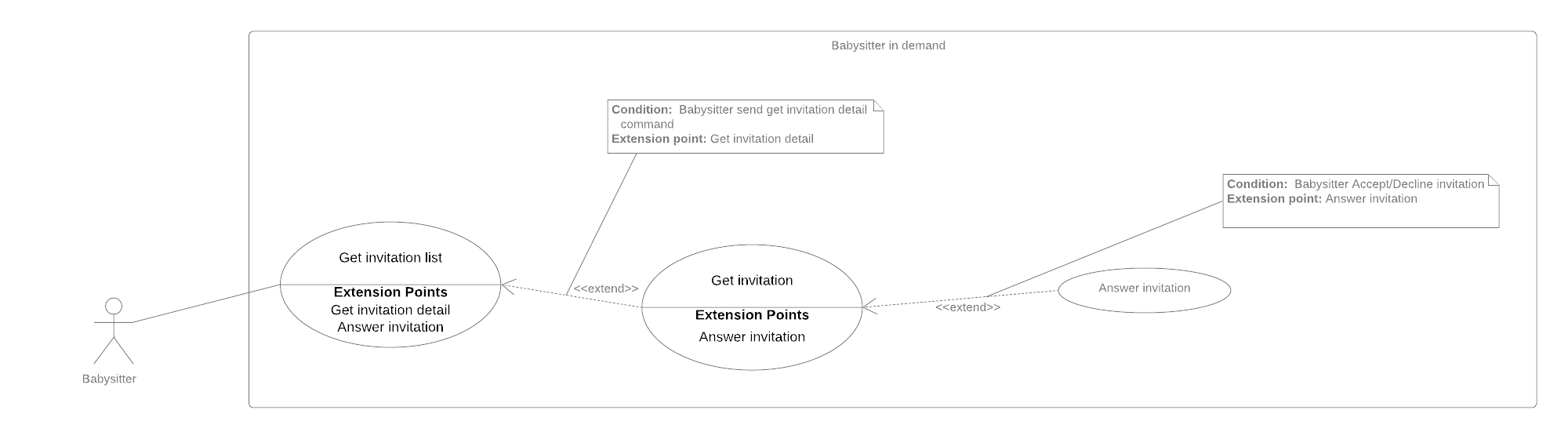


Figure 8 - <Babysitter> Answer invitation

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **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 18 - Use case BS.03 - <Babysitter> Answer invitation

##### <Babysitter> Get invitation

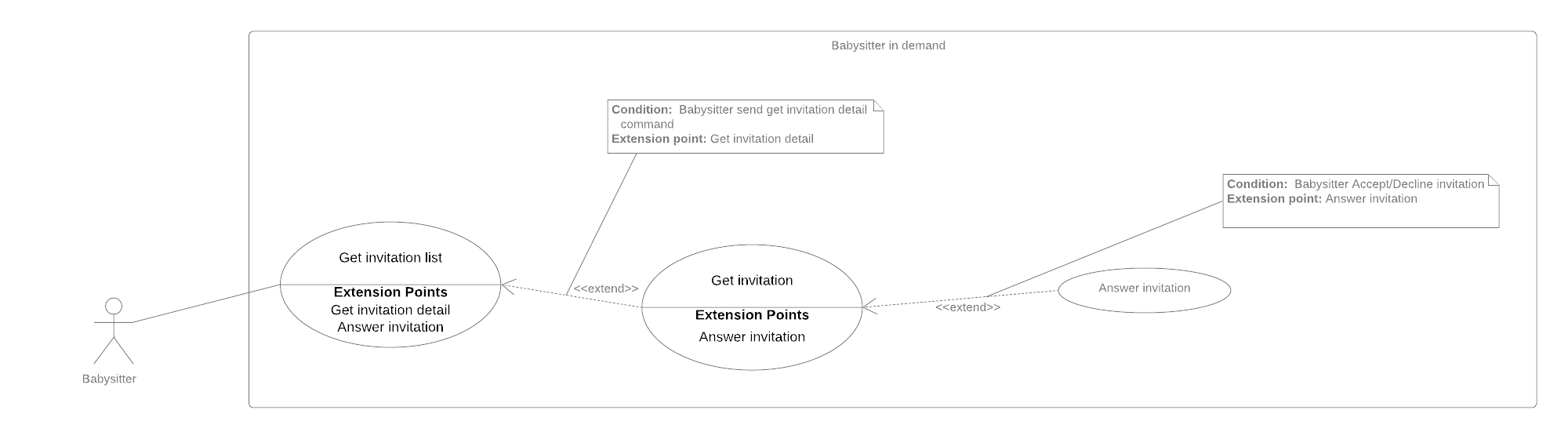


Figure 9 - <Babysitter>Get invitation

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **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 detail.   **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 19 - Use case BS.04 - <Babysitter> Get invitation

##### <Babysitter> Get invitation list

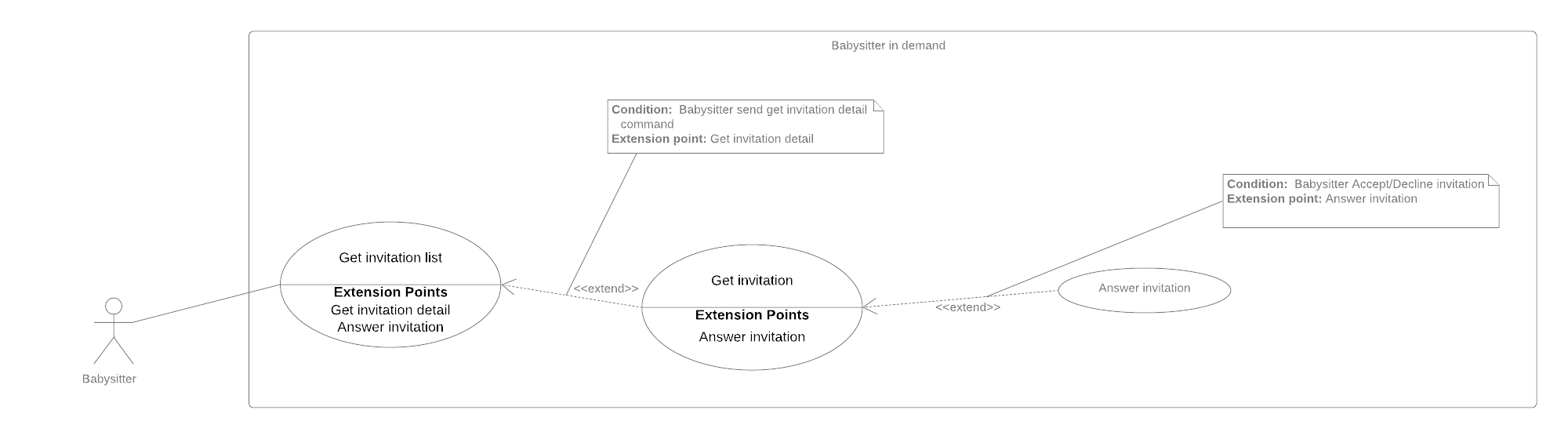


Figure 10 - <Babysitter> Get invitation list

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **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 20 - Use case BS.05 - <Babysitter> Get invitation list

##### <Babysitter> Get sitting history

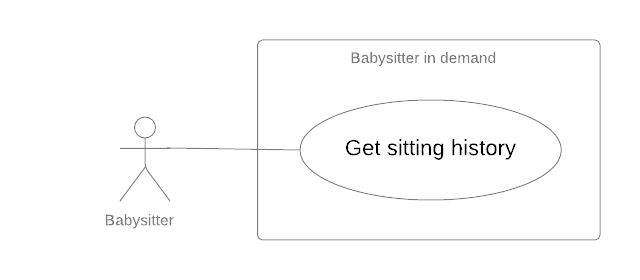


Figure 11 - <Babysitter> Get sitting history

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **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 21 - Use case BS.06 - <Babysitter> Get sitting history

#### <Parent> Overview Use Case

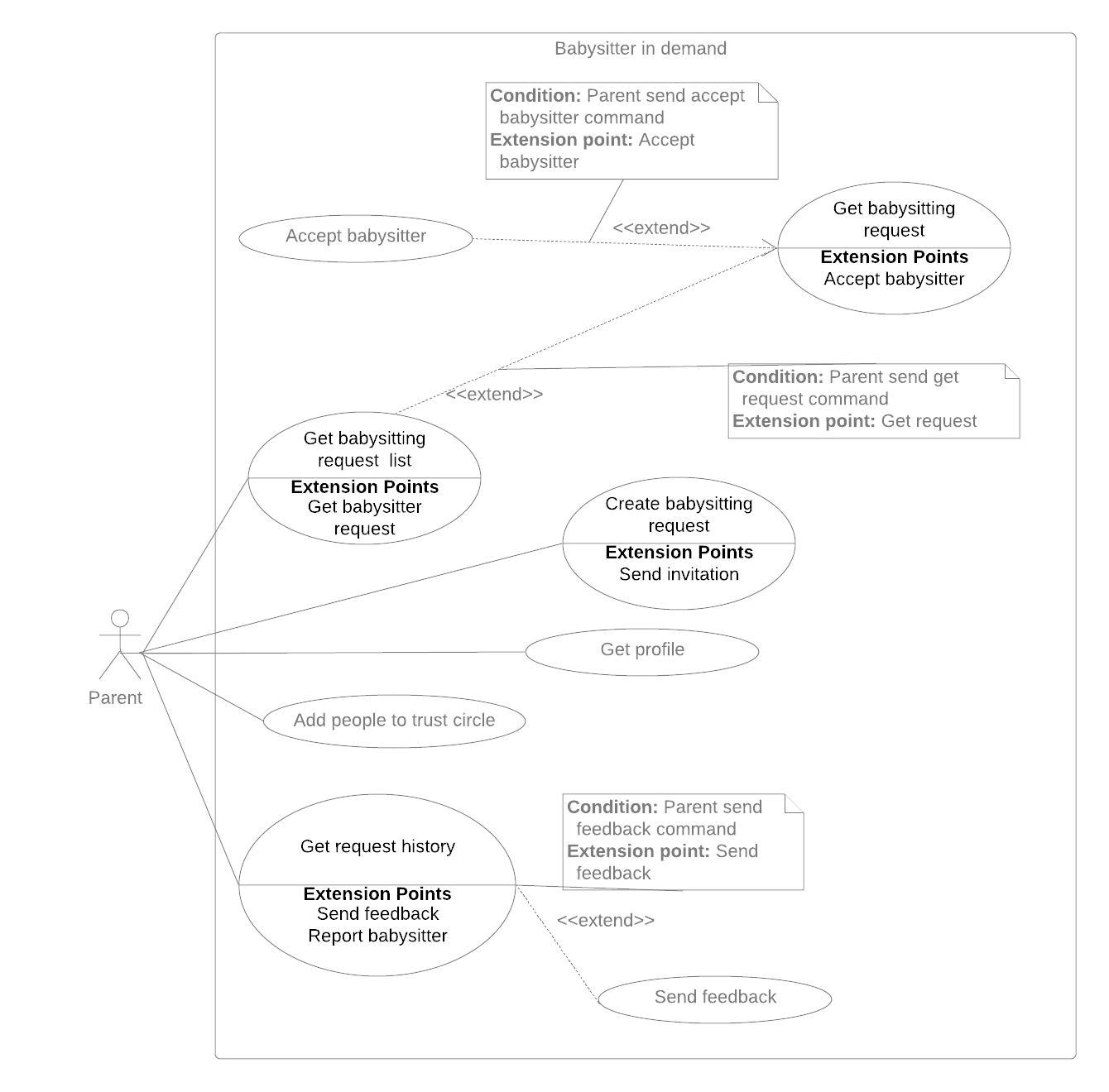


Figure 12 - <Parent> Overview Use Case

##### <Parent> Create babysitting request

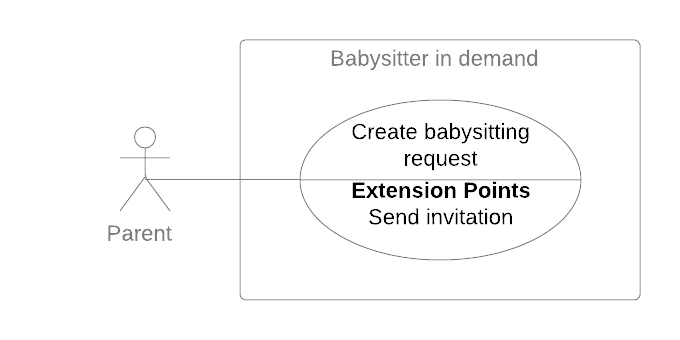


Figure 13 - <Parent> Create babysitting request

|  |  |  |  |
| --- | --- | --- | --- |
| **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 22 - Use case PR.01 - <Parent> Create babysitting request

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

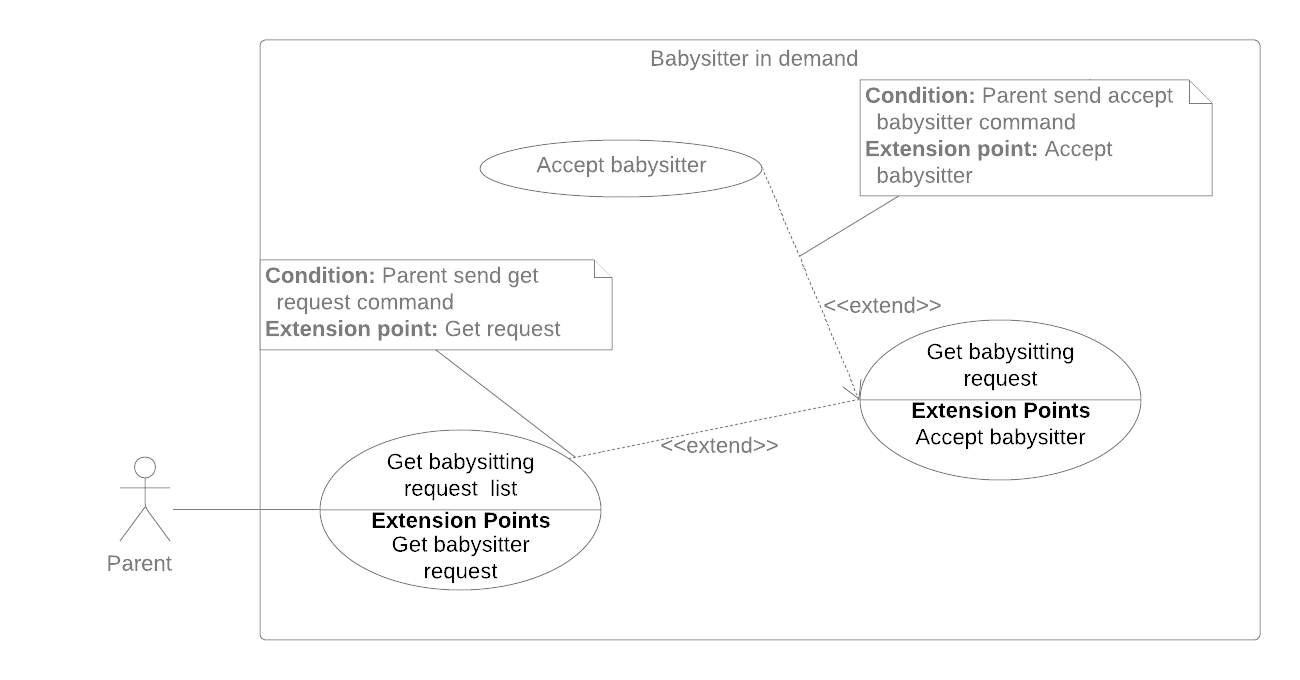


Figure 14 - <Parent> Get babysitting request list

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **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 23 - Use case PR.02 - <Parent> Get babysitting request list

##### <Parent> Get babysitting request

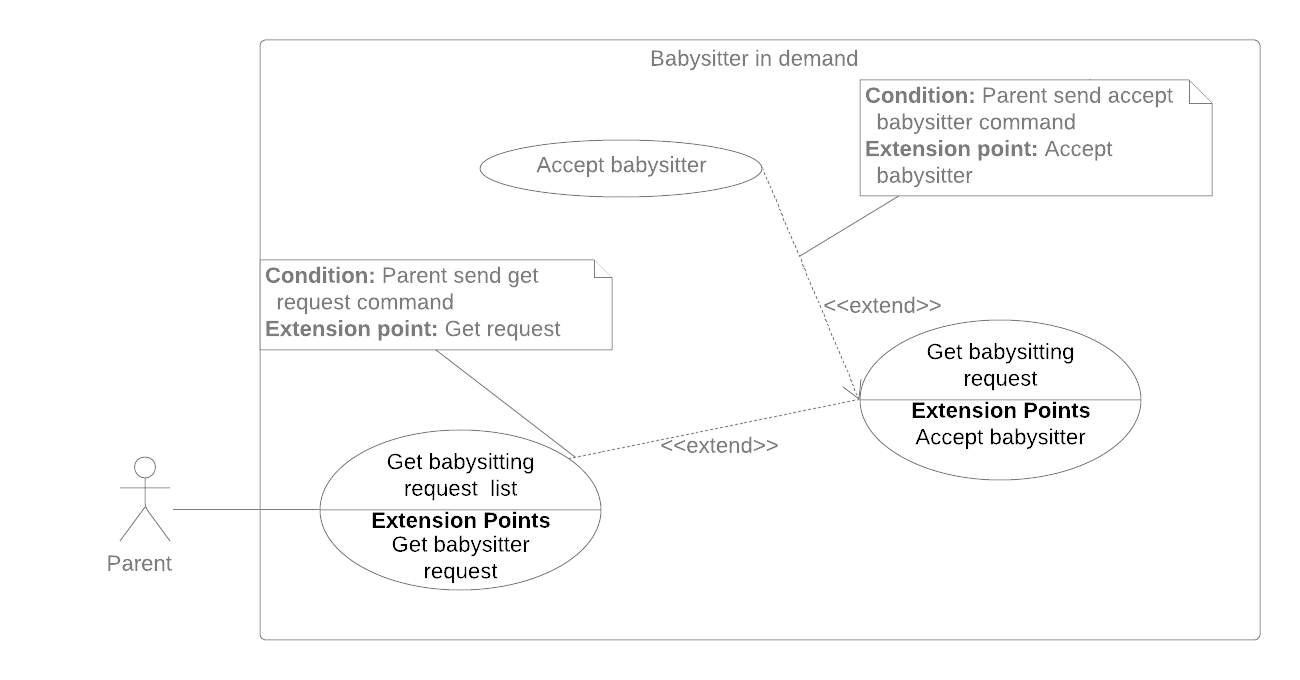


Figure 15 - <Parent> Get babysitting request

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **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 24 - Use case PR.03 - <Parent> Get babysitting request

##### <Parent> Accept babysitter

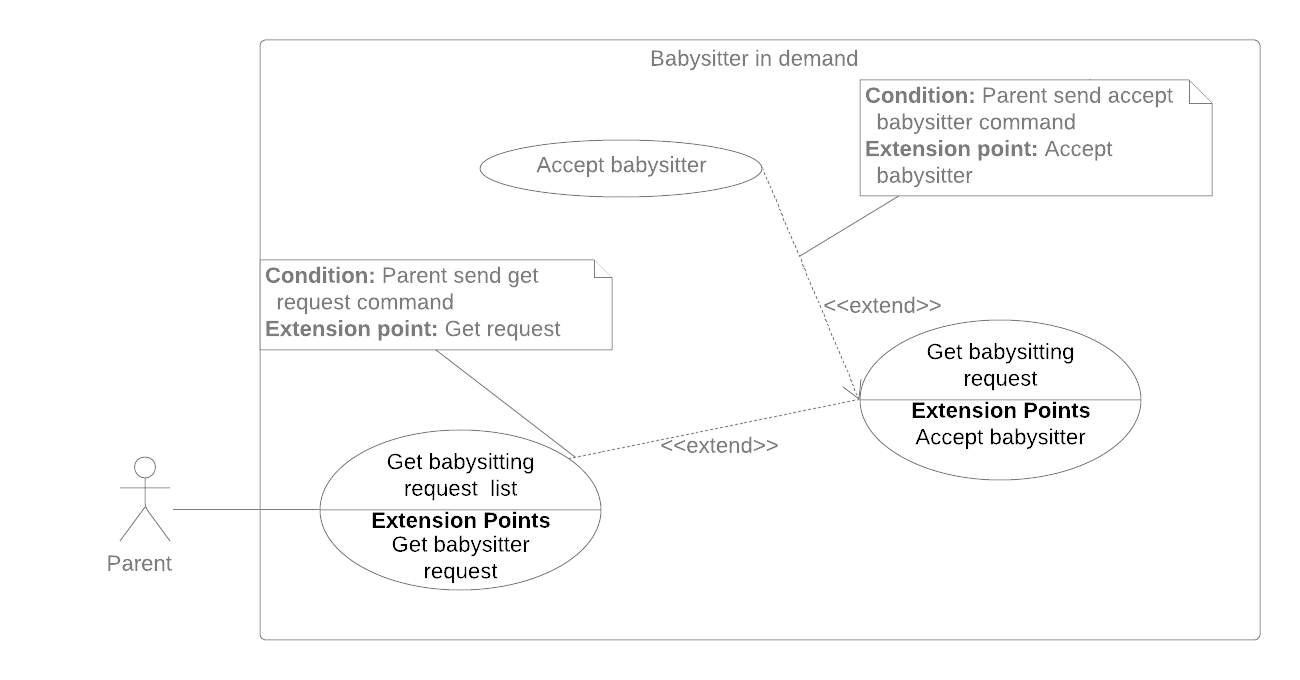


Figure 16 - <Parent> Accept babysitter

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **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 25 - Use case PR.04 - <Parent> Accept babysitter

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

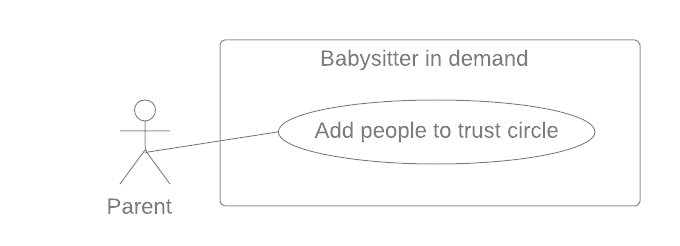


Figure 17 - <Parent> Add people to trust circle

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **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 26 - Use case PR.05 - <Parent> Add people to trust circle

##### <Parent> Get request history

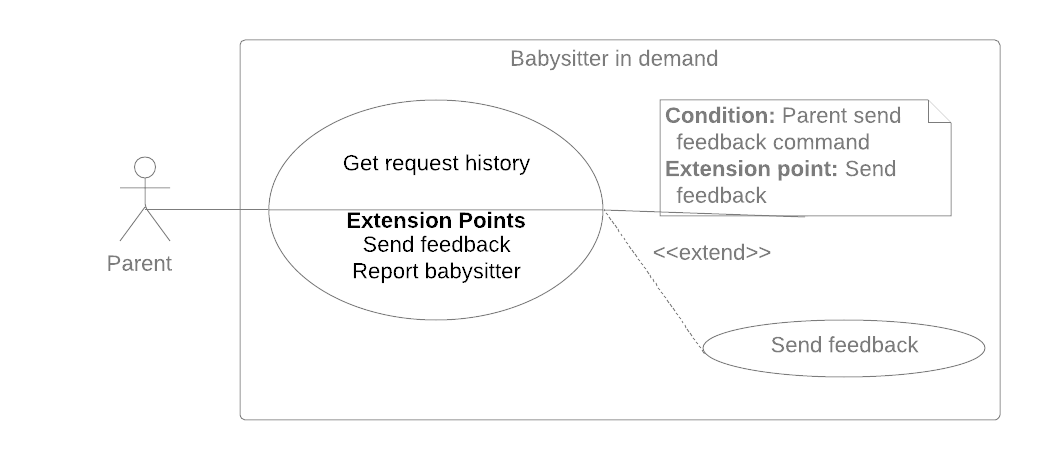


Figure 18 - <Parent> Get request history

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **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 27 - Use case PR.06 - <Parent> Get request history

##### <Parent> Get profile

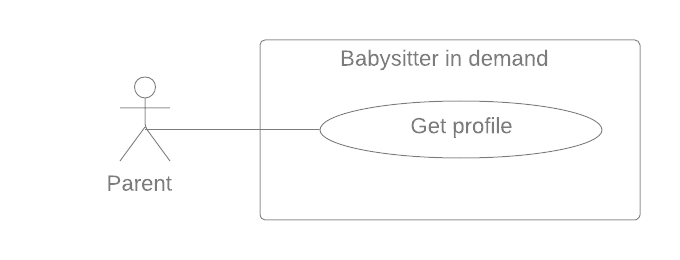


Figure 19 - <Parent> Get profile

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **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 28 - Use case PR.07 - <Parent> Get profile

##### <Parent> Send feedback

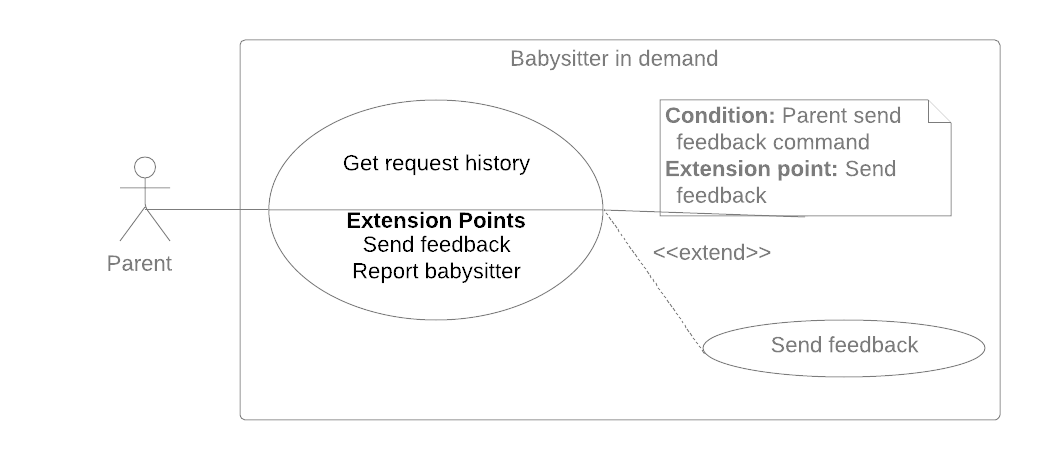


Figure 20 - <Parent> Send feedback

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **USE CASE – UC\_PR.08** | | | | |
| **Use Case No.** | UC\_PR.08 | **Use Case Version** | | 2.0 |
| **Use Case Name** | Send feedback | | | |
| **Author** | DongHN | | | |
| **Date** | 30/9/2019 | **Priority** | Normal | |
| **Actor:**   * Parent   **Summary:**   * This use case allows parent to send feedback for the sitting.   **Goal:**   * Parent can send feedback for the successful sitting.   **Triggers:**   * Parent send command to feedback for the sitting.   **Preconditions:**   * User must login into the system with Parent account. * The sitting must be success before this action.   **Post Conditions:**   * **Success**: * System records the feedback for the sitting. * **Fail**: * System shows error messages.   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | **Step** | **Actor Action** | **System Response** | | 1 | Parent sends command to create feedback form in request with status “Done “. | System returns the feedback form for the sitting. | | 2 | Parent input fields in the feedback form.  **[Alternative 1]** |  | | 3 | Parent sends command to save feedback form.  **[Alternative 2]** | System records feedback for the sitting from parent.  System shows success message. | | | | | |
| **Alternative Scenario:**  **[Alternative 1]**   |  |  |  | | --- | --- | --- | | **Step** | **Actor Action** | **System Response** | | 1 | Missing or wrong field defined. | System shows error message. | | | | | |
| **[Alternative 2]**   |  |  |  | | --- | --- | --- | | **Step** | **Actor Action** | **System Response** | | 1 | Parent sends command to cancel feedback form. | System back to Home Screen |   **Exceptions**:   |  |  |  | | --- | --- | --- | | **No** | **Cause** | **System Response** | | 1 | Failed to save feedback. | System shows error message. | | | | | |
| **Relationships:** N/A  **Business Rules:**   * A request with “Done” status will be allowed to feedback. * Information about parent feedback is always anonymous. * If a parent cancels a feedback, the parent can go back to history to feedback. | | | | |

Table 29 - Use case PR.08 - <Parent> Send feedback

#### <Staff> Overview Use Case

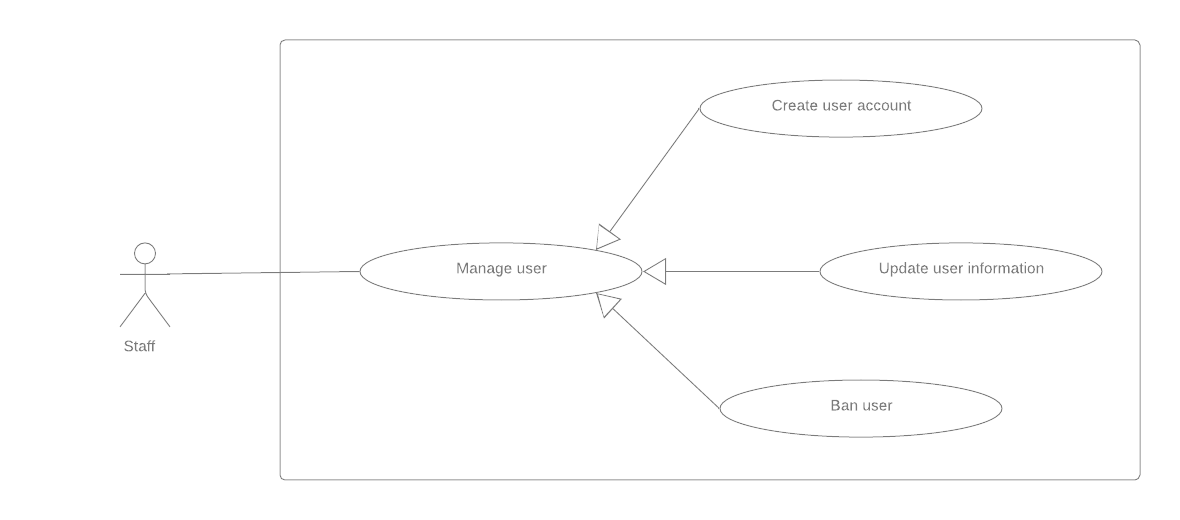


Figure 21 - <Staff> Overview Use Case



##### <Staff> Create user account

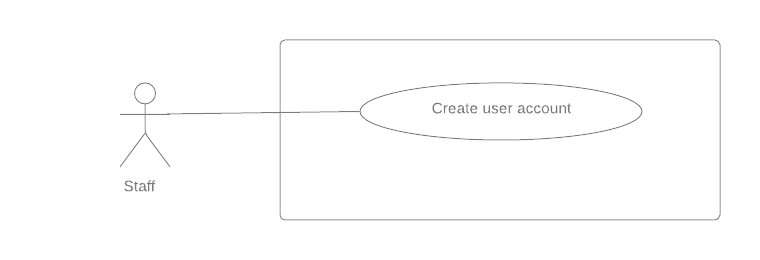


Figure 22 - <Staff> Create user account

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **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 30 - Use case ST.01 - <Staff> Create user account

##### <Staff> Update user information

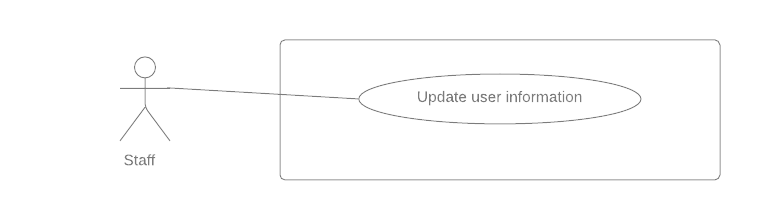


Figure 23 - <Staff> Update user information

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **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 31 - Use case ST.02 - <Staff> Update user information

##### <Staff> Ban user

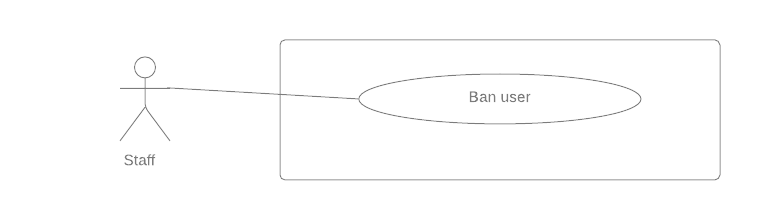


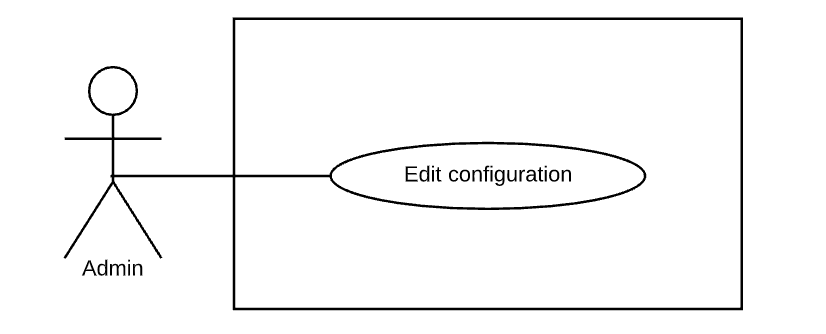
Figure 24 - <Staff> Ban user

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **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 32 - Use case ST.03 - <Staff> Ban user

##### <Staff> Edit configuration

Figure 25 - <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 33 - Use case ST.04 - <Staff> Edit configuration

#### <Payment> Pay

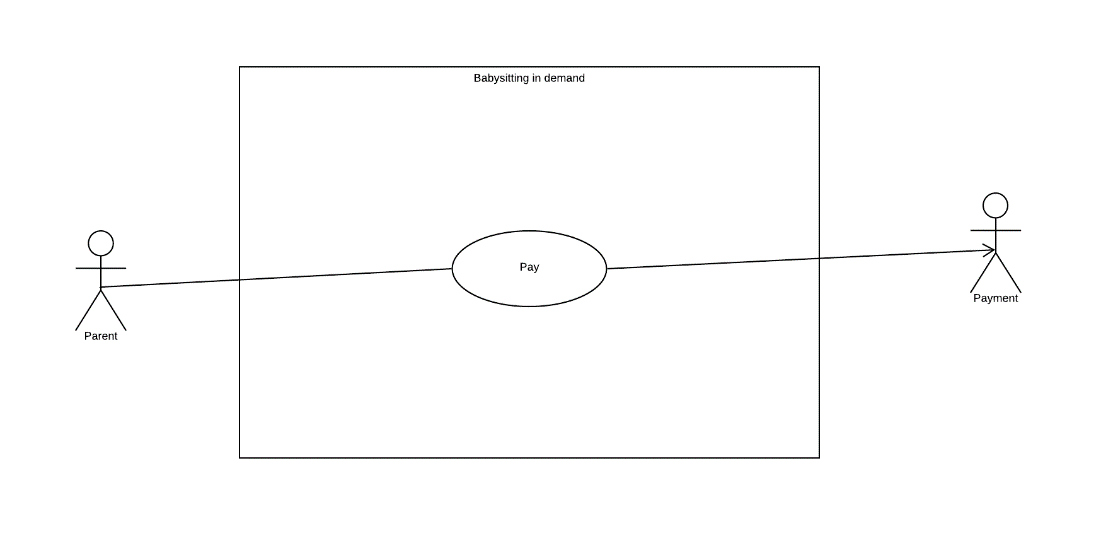


Figure 26 - <Payment> Pay

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **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 34 - Use case PM.01 - <Payment> Pay

##### <Payment> Refund

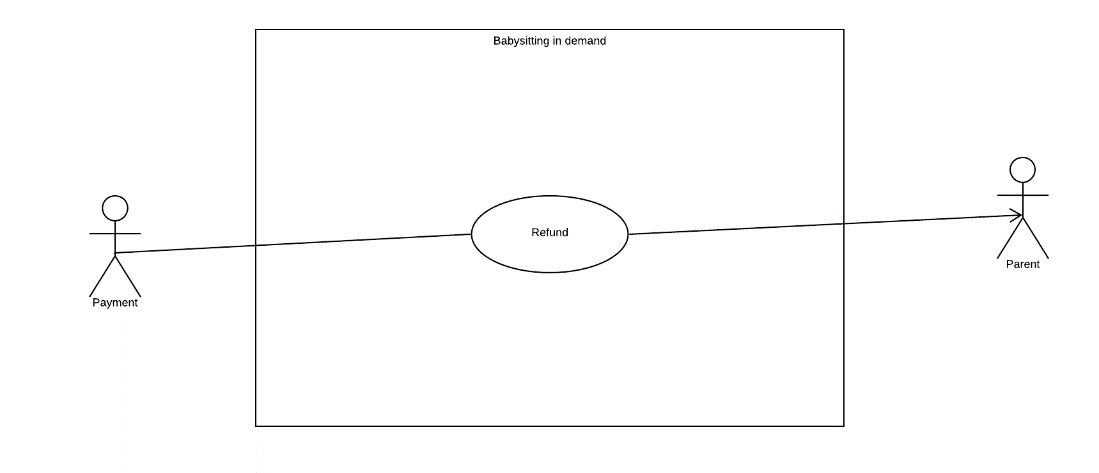


Figure 27 - <Payment> Refund

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **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 35 - Use case PM.02 - <Payment> Refund

#### <System Handler> Overview Use Case

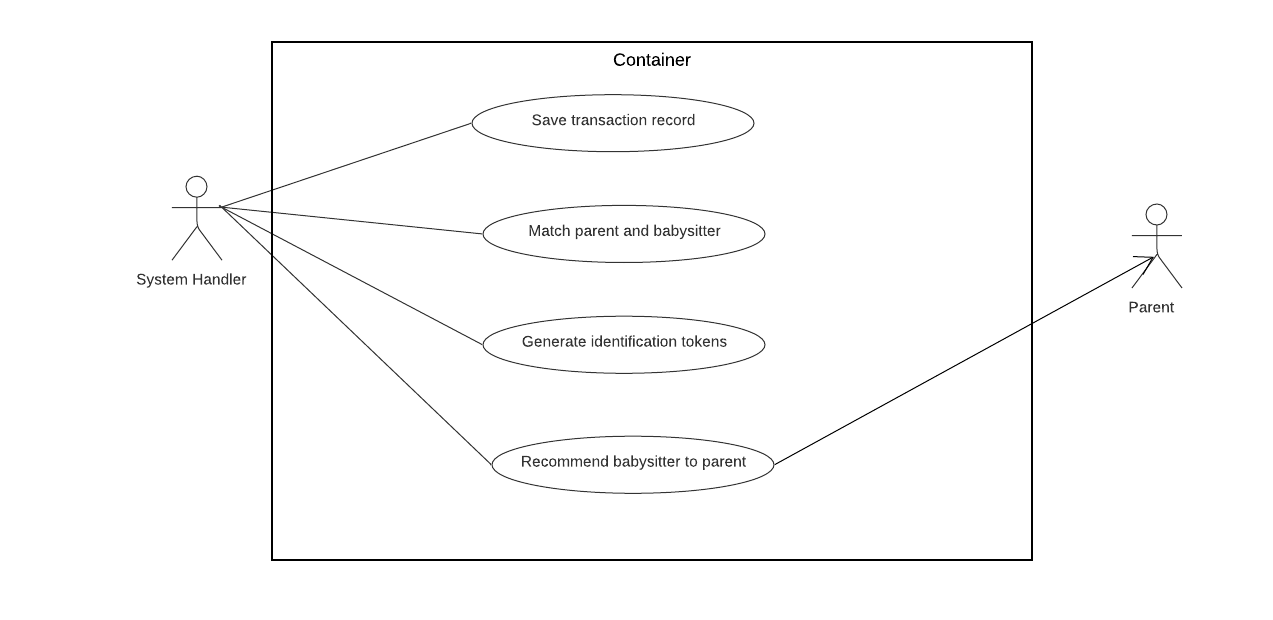


Figure 28 - <System Handler> Overview Use Case



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

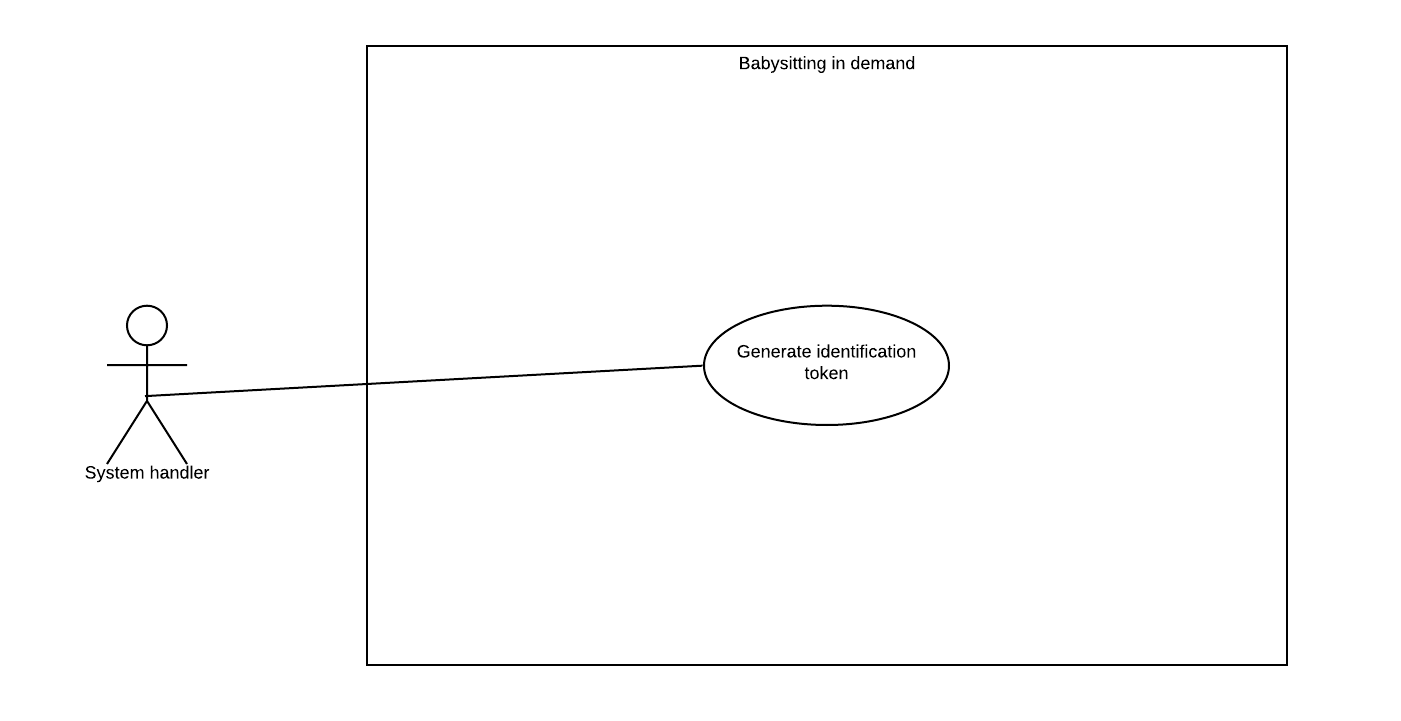


Figure 29 - <System Handler > Generate identification tokens

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **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 36 - Use case SH.01 - <System Handler > Generate identification tokens

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

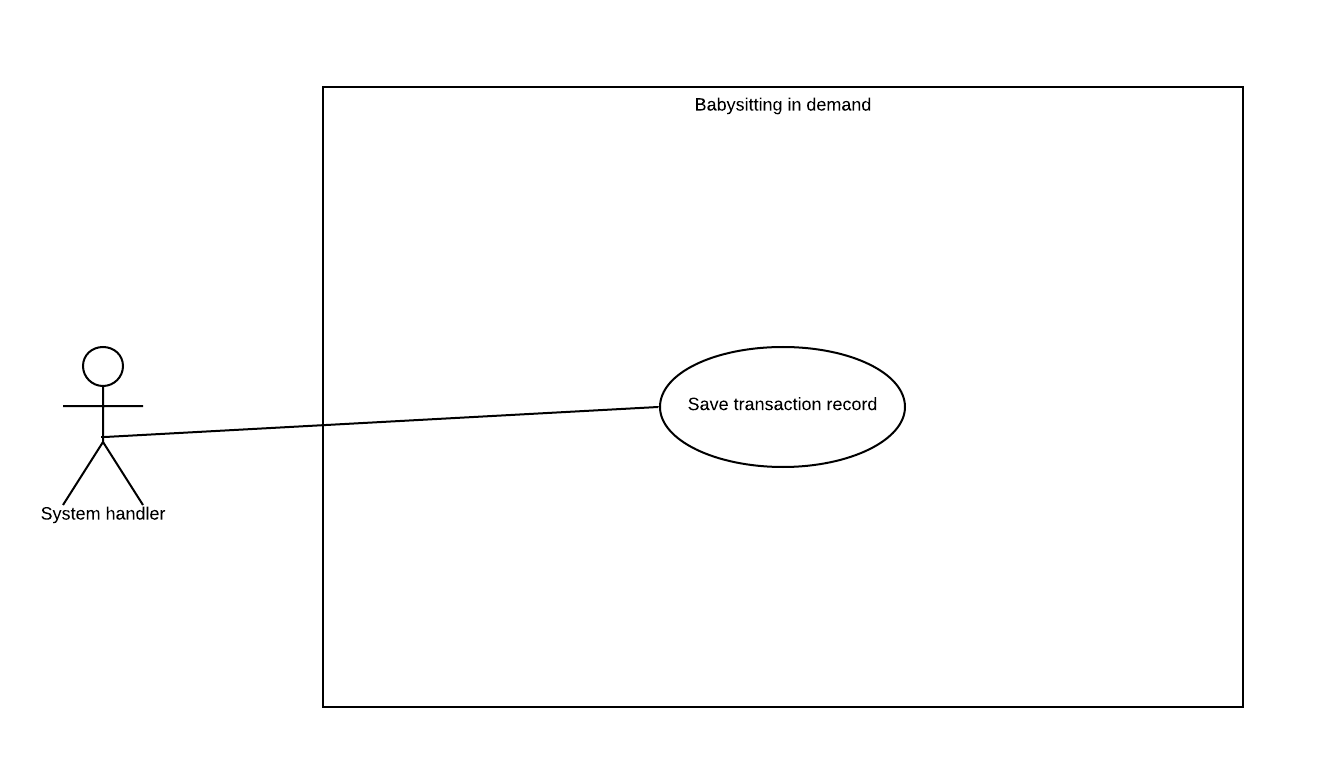


Figure 30 - <System Handler> Save transaction record

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **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 37 - Use case SH.02 - <System Handler> Save transaction record

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

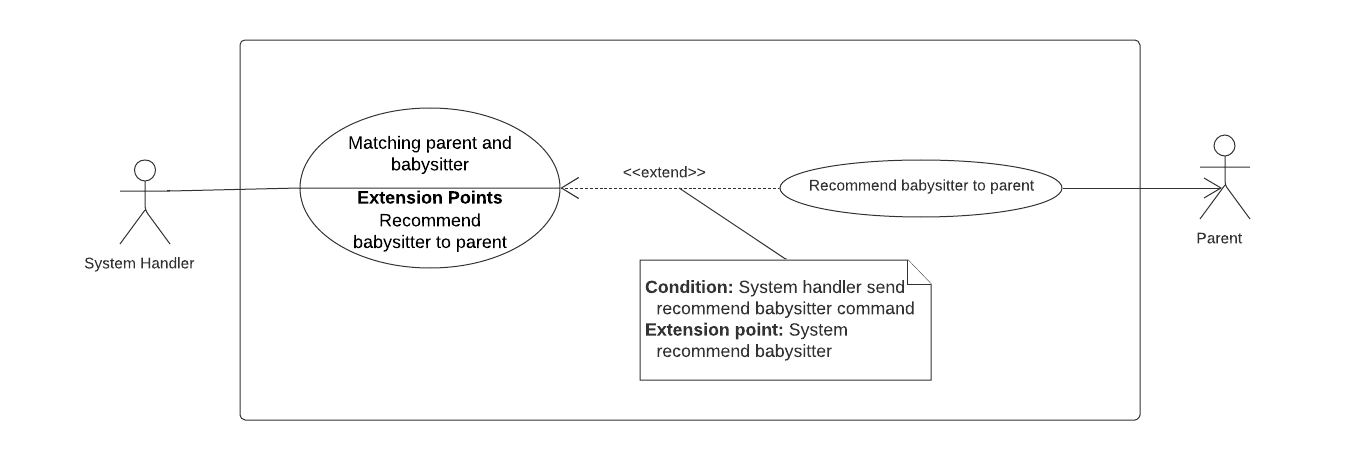


Figure 31 - <System Handler> Matching parent and babysitter

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **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 38 - Use case SH.03 - <System Handler> Matching parent and babysitter

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

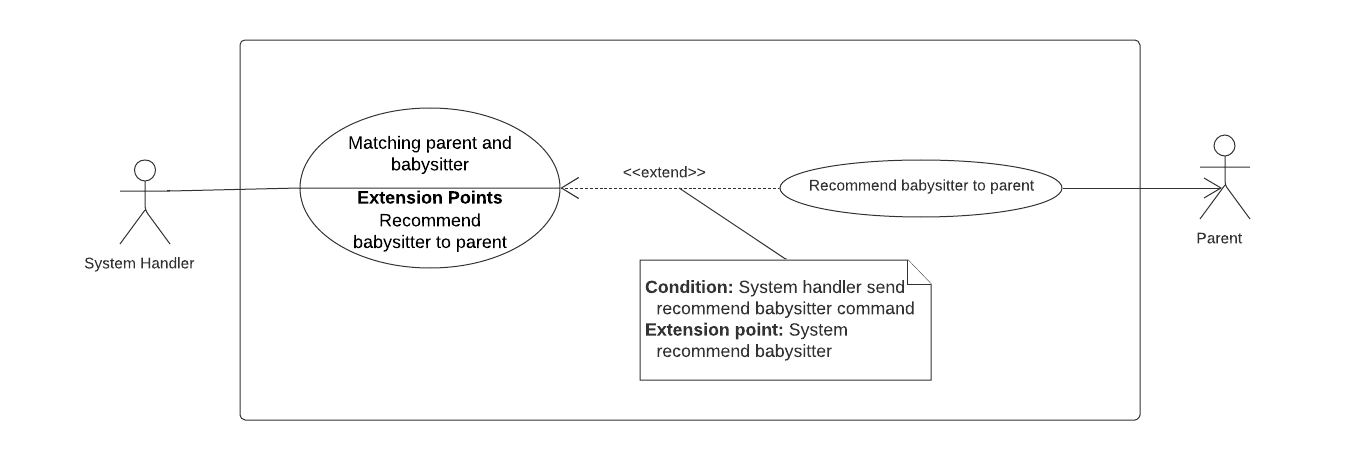


Figure 32 - <System Handler> Recommend babysitter to parent

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **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 39 - Use case SH.04 - <System Handler> Recommend babysitter to parent

## Software System Attribute

### Usability

* + System should be simple and easy to use

### Reliability

* + System’s functions should meet user’s needs
  + If a function fails, it should notify the user
  + System should be updated routine to meet reliability requirements

### Availability

* + System only available online

### Security

* + Each role of user has a specific permission to interact with system

### Maintainability

* + The system is divided into separate modules.

### Portability

* + The system can be used on iOS and Android

### Performance

* + Matching parent and babysitter should take less than 5 second to execute

## Conceptual Diagram

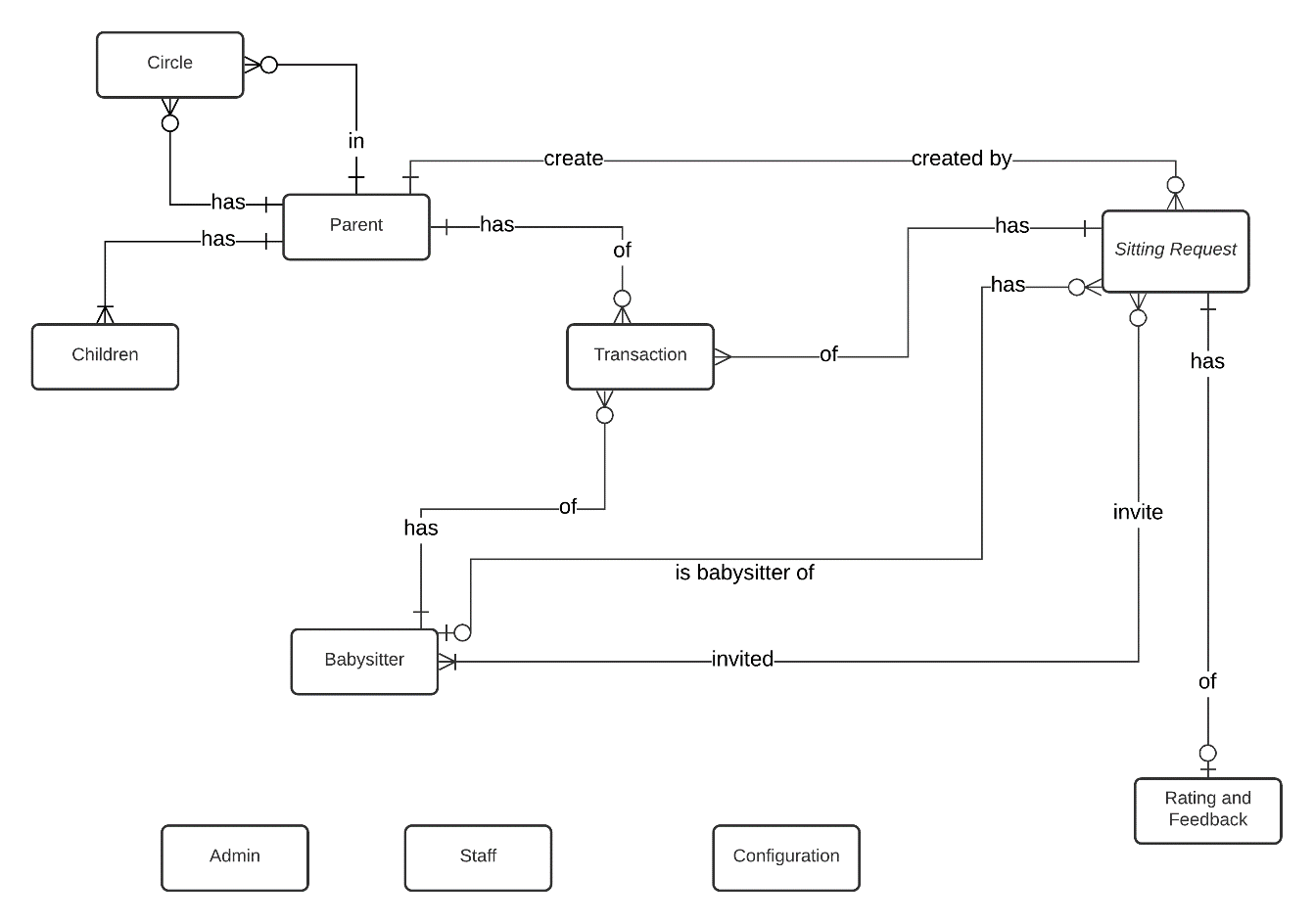


Figure 33 - Conceptual Diagram

**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 |

Table 40 - Data dictionary

# 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 Architectural Design

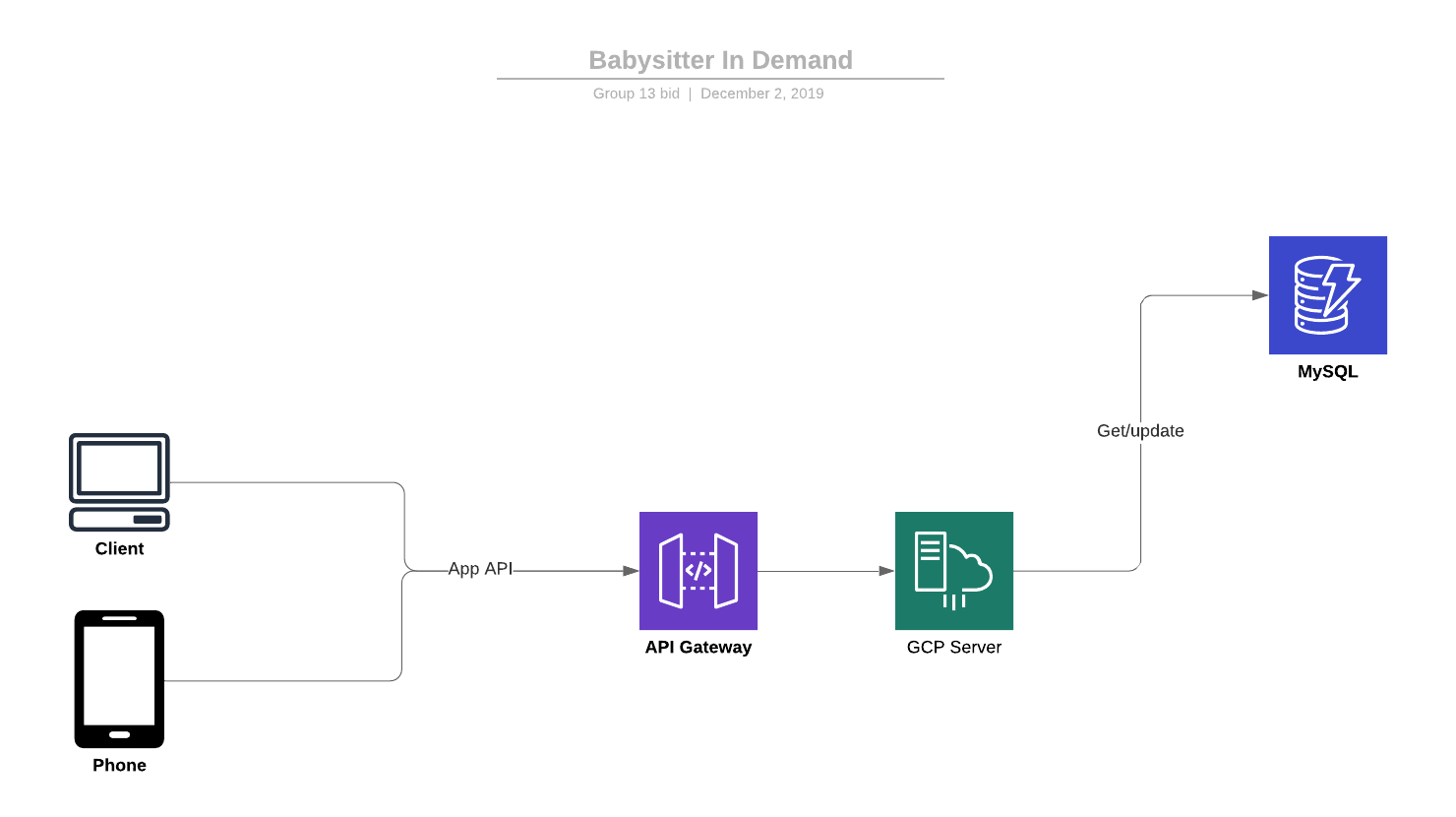


Figure 34 - System architecture design

## Component Diagram

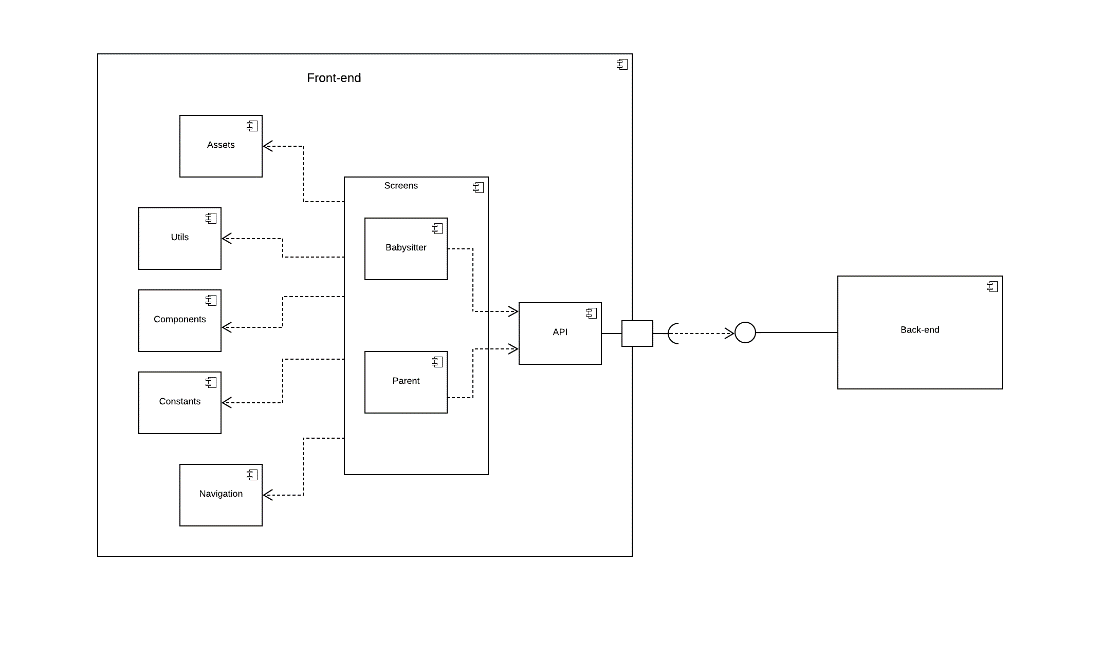


Figure 35 - Front-end component diagram

|  |  |
| --- | --- |
| **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 41 - Front-end component diagram dictionary

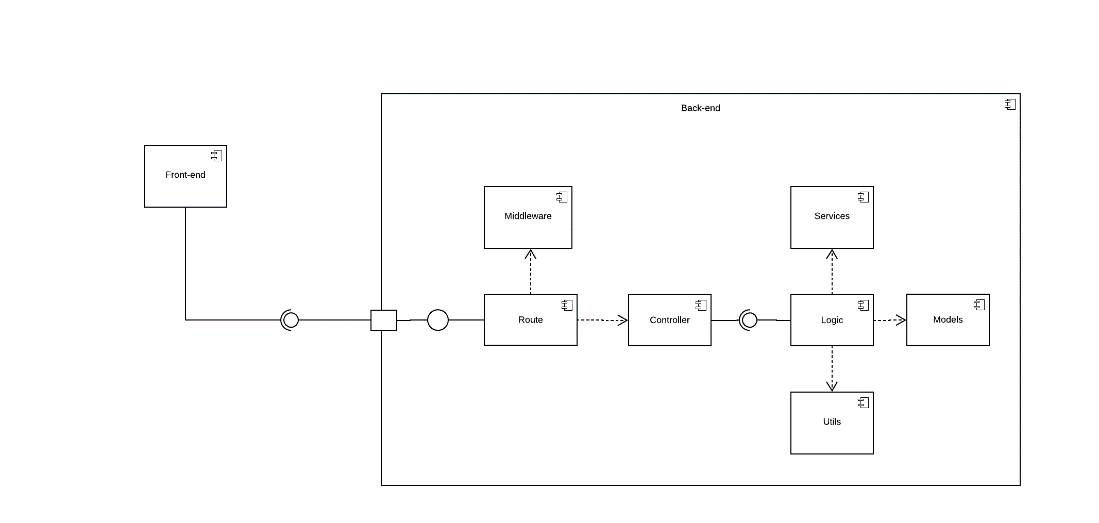


Figure 36 - Back-end component diagram

|  |  |
| --- | --- |
| **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 42 - Back-end component diagram dictionary

## Detail Description

### Class Diagram

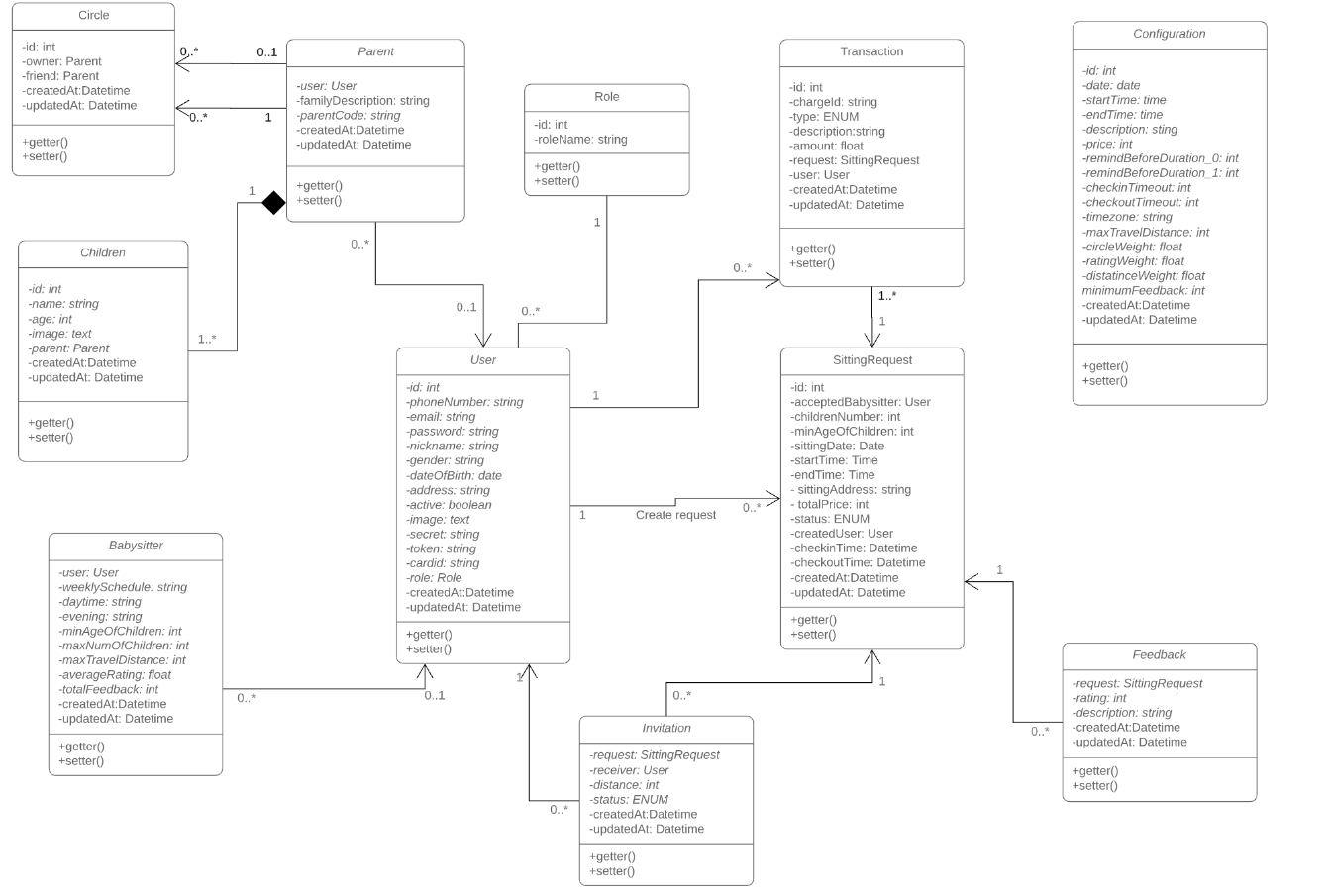


Figure 37 - Class Diagram

|  |  |  |
| --- | --- | --- |
| **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 43 - Class Dictionary

### Class Diagram Explanation

#### User

**Attribute**

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Visibility** | **Description** |
| id | Integer | private | Unique identifier of accounts |
| phoneNumber | String | private | Phone number of accounts |
| email | String | private | Email of accounts |
| password | String | private | Password of accounts |
| nickname | String | private | Nickname of accounts |
| gender | String | private | Gender of accounts |
| dateOfBirth | Date | private | Day of birth of accounts |
| address | String | private | Address of accounts |
| active | Boolean | private | Status of accounts |
| image | Text | private | Avatar of accounts |
| role | Role | private | Role of accounts |
|  |  |  |  |
| updatedAt | Datetime | private | Last update time of accounts |
| createdAt | Datetime | private | Create time of accounts |

Table 44 - User Attribute

**Method**

|  |  |  |  |
| --- | --- | --- | --- |
| **Method** | **Return type** | **Visibility** | **Description** |
| getter() | Attribute type | public | Get value of attributes |
| setter() | void | public | Set value for attributes |

Table 45 - User Method

#### Parent

**Attribute**

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Visibility** | **Description** |
| user | User | private | Account of parents |
| familyDescription | String | private | Description of parent’s family |
| parentCode | String | private | Code identifier for account in circle |
| updatedAt | Datetime | private | Last update time of accounts |
| createdAt | Datetime | private | Create time of accounts |

Table 46 -Parent Attribute

**Method**

|  |  |  |  |
| --- | --- | --- | --- |
| **Method** | **Return type** | **Visibility** | **Description** |
| getter() | Attribute type | public | Get value of attributes |
| setter() | void | public | Set value for attributes |

Table 47 - Parent Method

#### Children

**Attribute**

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Visibility** | **Description** |
| id | Integer | private | Unique identifier of children |
| name | String | private | Name of children |
| age | Integer | private | Age of children |
| image | Text | private | Avatar of children |
| parent | Parent | private | Parent which child belongs to |
| updatedAt | DateTime | private | Last update time of accounts |
| createdAt | Datetime | private | Create time of accounts |

Table 48 - Children Attribute

**Method**

|  |  |  |  |
| --- | --- | --- | --- |
| **Method** | **Return type** | **Visibility** | **Description** |
| getter() | Attribute type | public | Get value of attributes |
| setter() | void | public | Set value for attributes |

Table 49 - Children Method

#### Circle

**Attribute**

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Visibility** | **Description** |
| id | Integer | private | Unique identifier of circle |
| owner | Parent | private | Circle’s owner |
| friend | Parent | private | List of parent in circle |
| updatedAt | Datetime | private | Last update time of accounts |
| createdAt | Datetime | private | Create time of accounts |

Table 50 -Circle Attribute

**Method**

|  |  |  |  |
| --- | --- | --- | --- |
| **Method** | **Return type** | **Visibility** | **Description** |
| getter() | Attribute type | public | Get value of attributes |
| setter() | void | public | Set value for attributes |

Table 51 -Circle Method

#### Babysitter

**Attribute**

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Visibility** | **Description** |
| user | User | private | Account of babysitters |
| weeklySchedule | String | private | Working date of babysitters |
| daytime | String | private | Working’s daytime of babysitter |
| evening | String | private | Working’s evening time of babysitters |
| minAgeOfChildren | Interger | private | Minimum children’s age babysitter can handle in a sitting |
| maxNumOfChildren | Integer | private | Maximum number of children babysitter can handle in a sitting |
| maxTravelDistance | Integer | private | Maximum travel distance babysitter can go for the sitting |
| averageRating | Float | private | Average rating of babysitter for all sitting |
| totalFeedback | Integer | private | Total feedbacks of babysitters |
| updatedAt | Datetime | private | Last update time of accounts |
| createdAt | Datetime | private | Create time of accounts |

Table 52 -Babysitter Attribute

**Method**

|  |  |  |  |
| --- | --- | --- | --- |
| **Method** | **Return type** | **Visibility** | **Description** |
| getter() | Attribute type | public | Get value of attributes |
| setter() | void | public | Set value for attributes |

Table 53 - Babysitter Method

#### Role

**Attribute**

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Visibility** | **Description** |
| id | Integer | private | Unique identifier of roles |
| roleName | String | private | Name of roles |

Table 54 -Role Attribute

**Method**

|  |  |  |  |
| --- | --- | --- | --- |
| **Method** | **Return type** | **Visibility** | **Description** |
| getter() | Attribute type | public | Get value of attributes |
| setter() | void | public | Set value for attributes |

Table 55 -Role Method

#### SittingRequest

**Attribute**

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Visibility** | **Description** |
| id | Integer | private | Unique identifier of sitting requests |
| acceptedBabysitter | User | private | Assigned user work for the sitting |
| childrenNumber | Integer | private | Number of children in the sitting |
| sittingDate | Date | private | Date of sittings |
| startTime | Time | private | Start time of sittings |
| endTime | Time | private | End time of sittings |
| totalPrice | Integer | private | Price of sittings |
| address | String | private | Address of sittings |
| status | ENUM | private | Status of sittings |
| createdUser | User | private | Created parent user of sittings |
| updatedAt | Datetime | private | Last update time of accounts |
| createdAt | Datetime | private | Create time of accounts |

Table 56 -SittingRequest Attribute

**Method**

|  |  |  |  |
| --- | --- | --- | --- |
| **Method** | **Return type** | **Visibility** | **Description** |
| getter() | Attribute type | public | Get value of attributes |
| setter() | void | public | Set value for attributes |

Table 57 – SittingRequest Method

#### Feedback

**Attribute**

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Visibility** | **Description** |
| request | SittingRequest | private | Sitting request of feedbacks |
| rating | Integer | private | Mark of feedbacks |
| description | String | private | Description of feedbacks |
| updatedAt | Datetime | private | Last update time of accounts |
| createdAt | Datetime | private | Create time of accounts |

Table 58 -Feedback Attribute

**Method**

|  |  |  |  |
| --- | --- | --- | --- |
| **Method** | **Return type** | **Visibility** | **Description** |
| getter() | Attribute type | public | Get value of attributes |
| setter() | void | public | Set value for attributes |

Table 59 -Feedback Method

#### Invitation

**Attribute**

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Visibility** | **Description** |
| request | SittingRequest | private | Sitting request of invitations |
| receiver | User | private | Receiver of invitations |
| distance | Integer | private | Distance between sitting’s address and babysitter’s address |
| status | ENUM | private | Status of invitations |
| updatedAt | Datetime | private | Last update time of accounts |
| createdAt | Datetime | private | Create time of accounts |

Table 60 -Invitation Attribute

**Method**

|  |  |  |  |
| --- | --- | --- | --- |
| **Method** | **Return type** | **Visibility** | **Description** |
| getter() | Attribute type | public | Get value of attributes |
| setter() | void | public | Set value for attributes |

Table 61 - Invitation Method

#### Configuration

**Attribute**

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Visibility** | **Description** |
| id | Integer | private | Unique identifier of configurations |
| date | Date | private | Date of configurations |
| startTime | Time | private | Start time of configurations |
| endTime | Time | private | End time of configurations |
| description | String | private | Description of configurations |
| price | Integer | private | Price setting |
| remindBeforeDuration\_0 | Integer | private | Duration from remind time to sitting time |
| remindBeforeDuration\_1 | Integer | private | Duration from remind time to sitting time |
| checkinTimeout | Integer | private | Timeout of check-in time from start time of sitting request |
| checkoutTimeout | Integer | private | Timeout of check-out time from end time of sitting request |
| timezone | String | private | Timezone of system |
| maxTravelDistance | Integer | private | Distance setting for recommend |
| circleWeight | Float | private | Circle weight setting |
| ratingWeight | Float | private | Rating weight setting |
| distanceWeight | Float | private | Distance weight setting |
| minimumFeedback | Integer | private | Minimum feedback setting |
| updatedAt | DateTime | private | Last update time of accounts |
| createdAt | Datetime | private | Create time of accounts |

Table 62 -Configuration Attribute

**Method**

|  |  |  |  |
| --- | --- | --- | --- |
| **Method** | **Return type** | **Visibility** | **Description** |
| getter() | Attribute type | public | Get value of attributes |
| setter() | void | public | Set value for attributes |

Table 63 - Configuration Method

#### Transaction

**Attribute**

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Visibility** | **Description** |
| id | Integer | private | Unique identifier of transactions |
| chargeId | String | private | Billing id of transactions |
| type | ENUM | private | Type of transactions |
| description | String | private | Description of transactions |
| amount | Float | private | Amount of transactions |
| request | SittingRequest | private | Sitting request of transactions |
| user | User | private | Created user of transactions |
| updatedAt | Datetime | private | Last update time of accounts |
| createdAt | Datetime | private | Create time of accounts |

Table 64 -Transaction Attribute

**Method**

|  |  |  |  |
| --- | --- | --- | --- |
| **Method** | **Return type** | **Visibility** | **Description** |
| getter() | Attribute type | public | Get value of attributes |
| setter() | void | public | Set value for attributes |

Table 65 -Transaction Method

### Interaction Diagram

#### <Guest> Login

Summary: This diagram shows the process of verifying the user

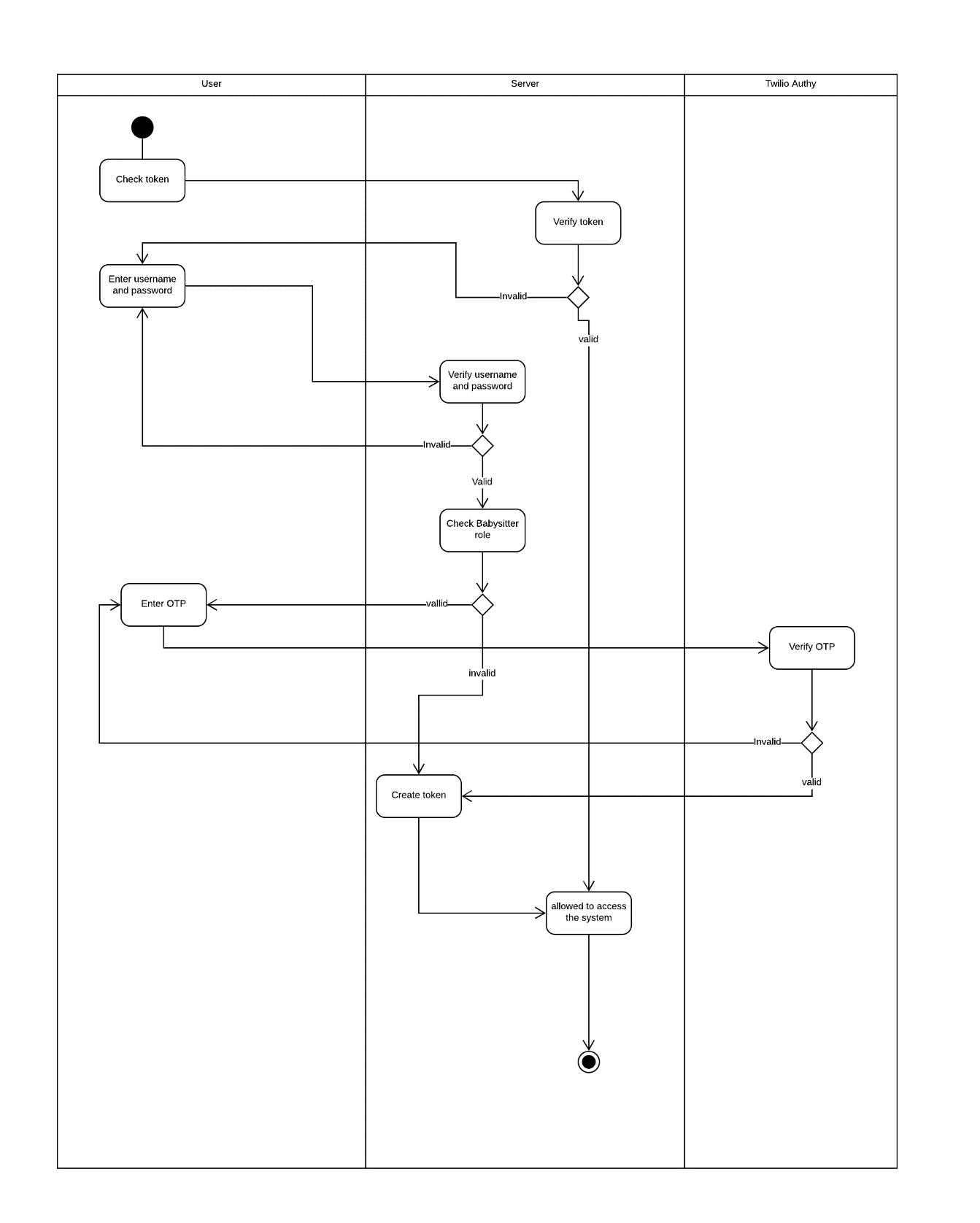


Figure 38 - <Guest> Login

#### <Babysitter> Answer invitation

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

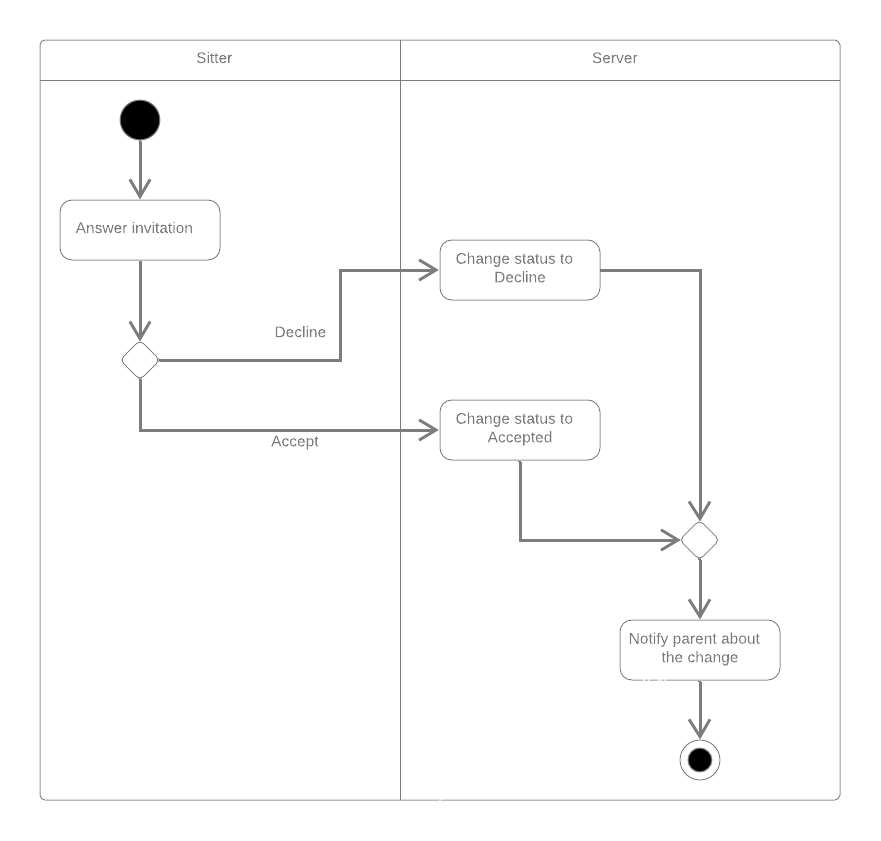


Figure 39 - <Babysitter> Answer invitation

#### <Parent> Create babysitting request

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

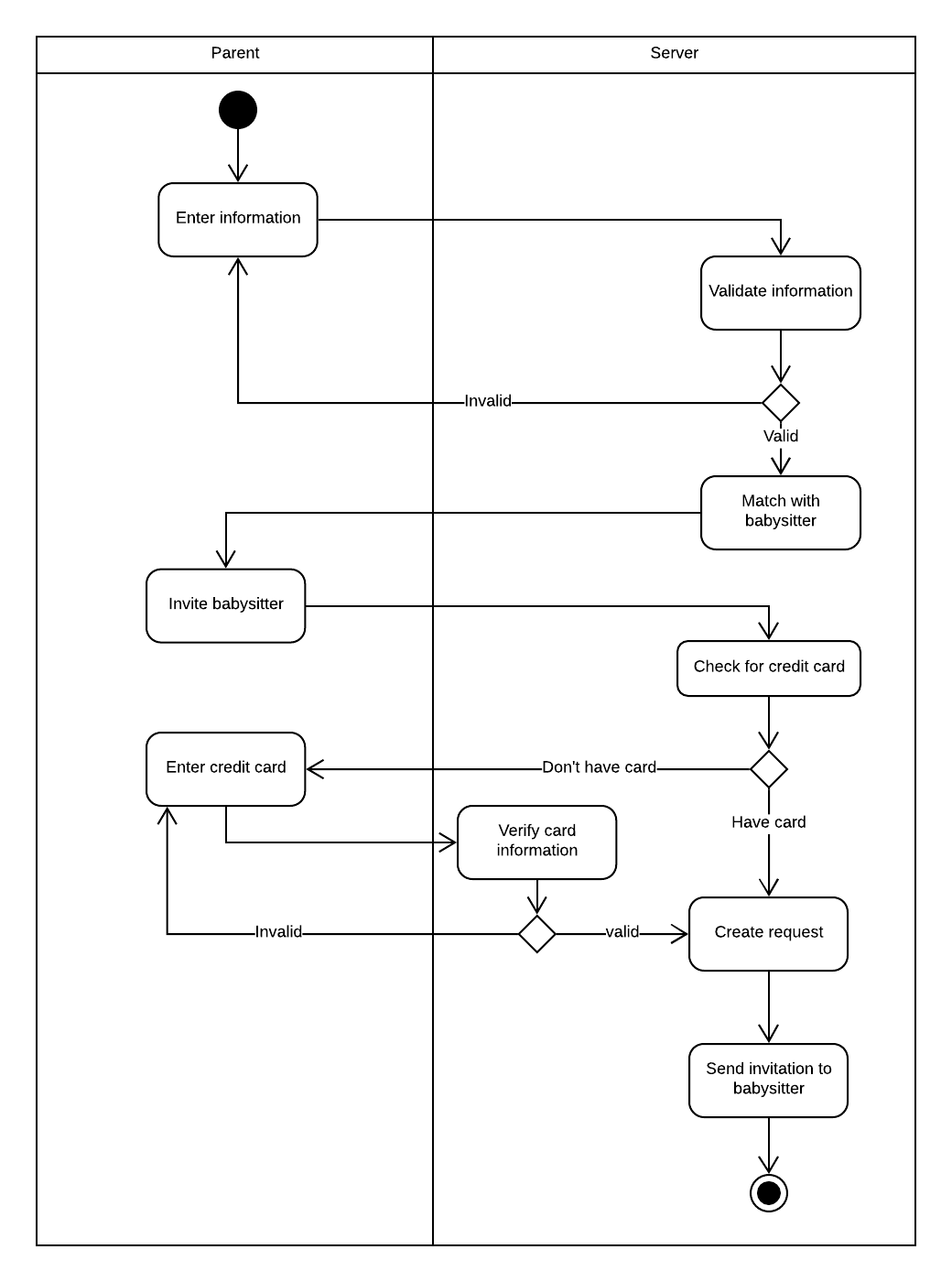


Figure 40: <Parent> Create babysitting request

#### <Parent> Accept babysitter

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

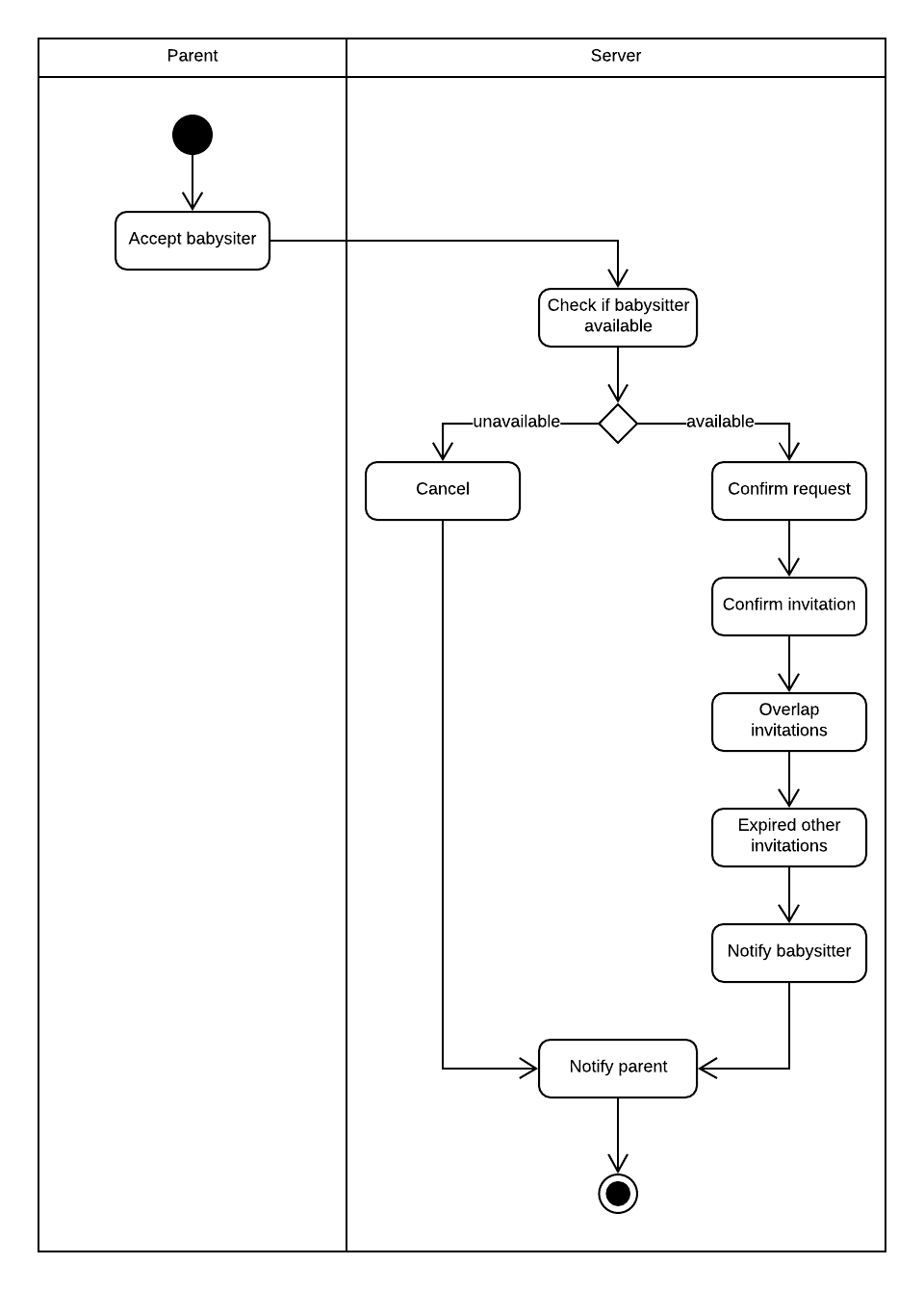


Figure 41 - <Parent> Accept babysitter

#### <Parent> Send feedback

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

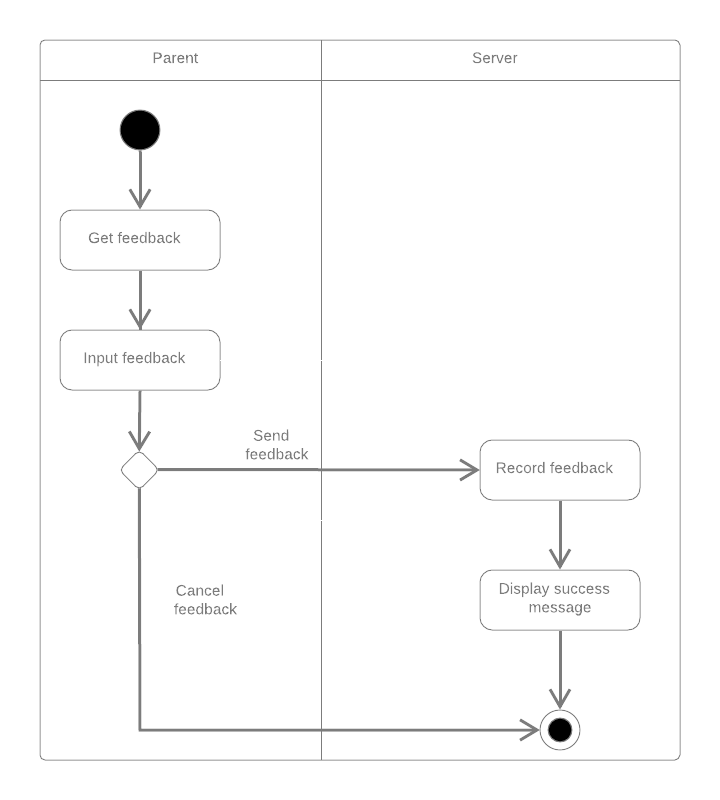


Figure 42 - <Parent> Send feedback

#### <Payment> Pay

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

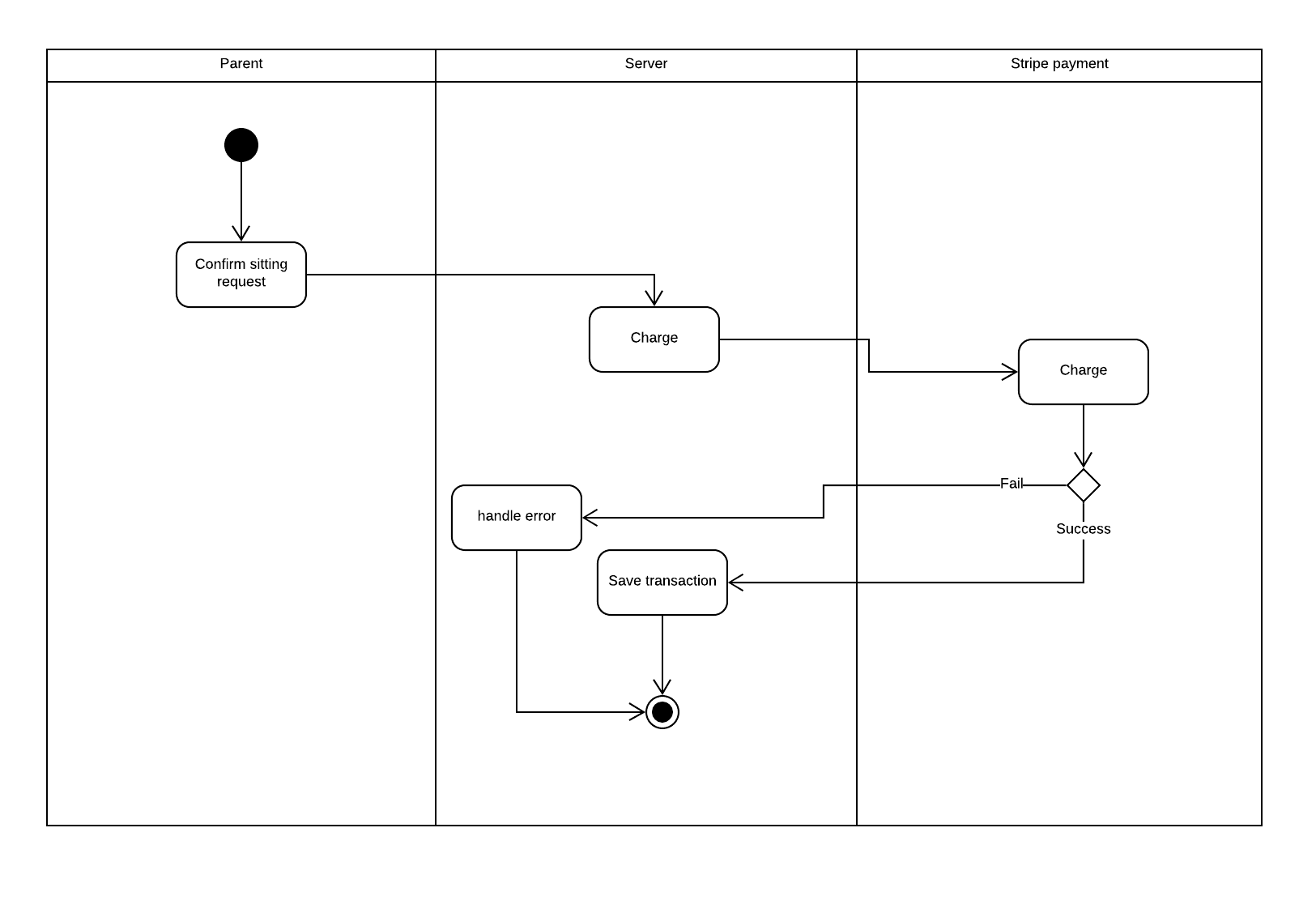


Figure 43 - <Payment> Pay

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

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

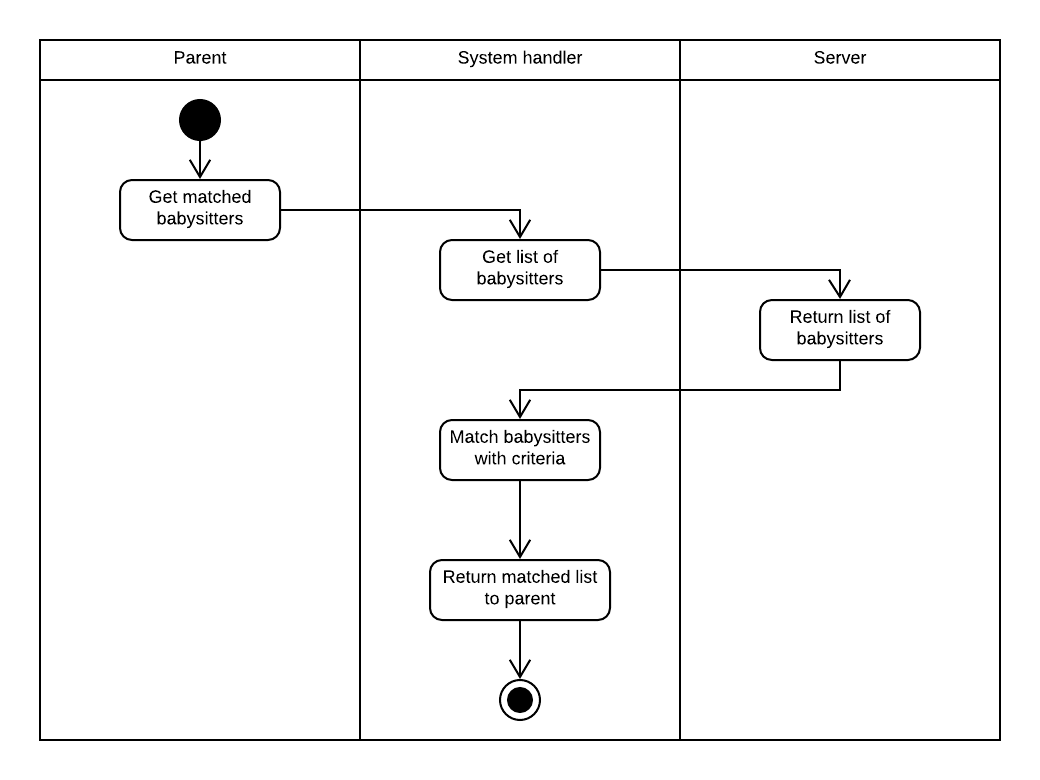


Figure 44 - <System Handler> Match parent and babysitter

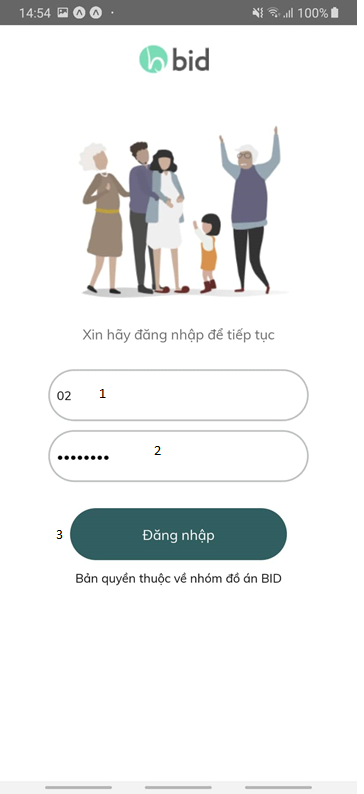
## Interface

### Mobile application design

#### Parent and Sitter login screen

2

1



**Fields**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **No** | **Field Name** | **Description** | **Read only** | **Mandatory** | **Control type** | **Data type** | **Length** |
| 1 | input phone | User enter phone number | No | Yes | Text input | text | N/A |
| 2 | Input password | User enter password | No | Yes | password | password | N/A |

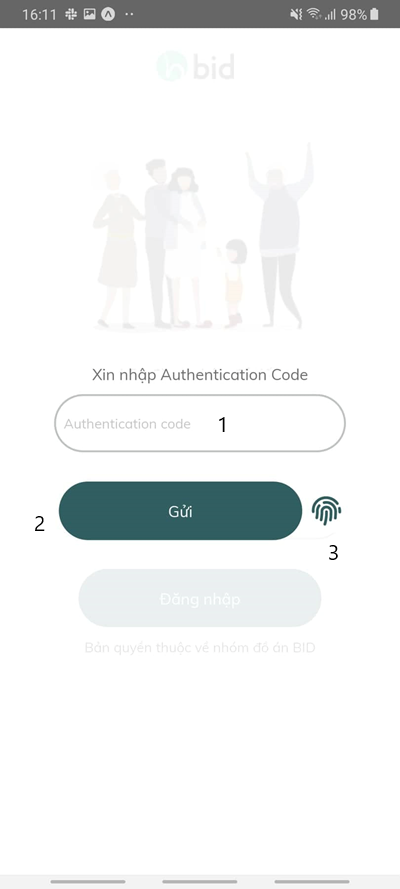
**Buttons/Hyperlink**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No** | **Function** | **Description** | **Validation** | **Outcome** |
| 3 | loginByPhone | Login to application by using phone number | N/A | Transfer to OTP popup |

##### Input OTP popup

2

3



**Fields**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **No** | **Field Name** | **Description** | | **Read only** | | **Mandatory** | | **Control type** | | **Data type** | | **Length** | |
| 1 | Input OTP | | User enter OTP | | No | | Yes | | Input number | | number | | N/A | |

**Buttons/Hyperlink**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No** | **Function** | **Description** | **Validation** | **Outcome** |
| 2 | sendLoginOTP | Login to application by using OTP | N/A | Transfer to Sitter home screen when account is babysitter  Transfer to Parent home screen when account is parent |
| 3 | loginByFinger | Login to application by using finger | N/A | Transfer to Sitter home screen when account is babysitter  Transfer to Parent home screen when account is parent |

#### Parent Interface design

##### Parent Home screen

1

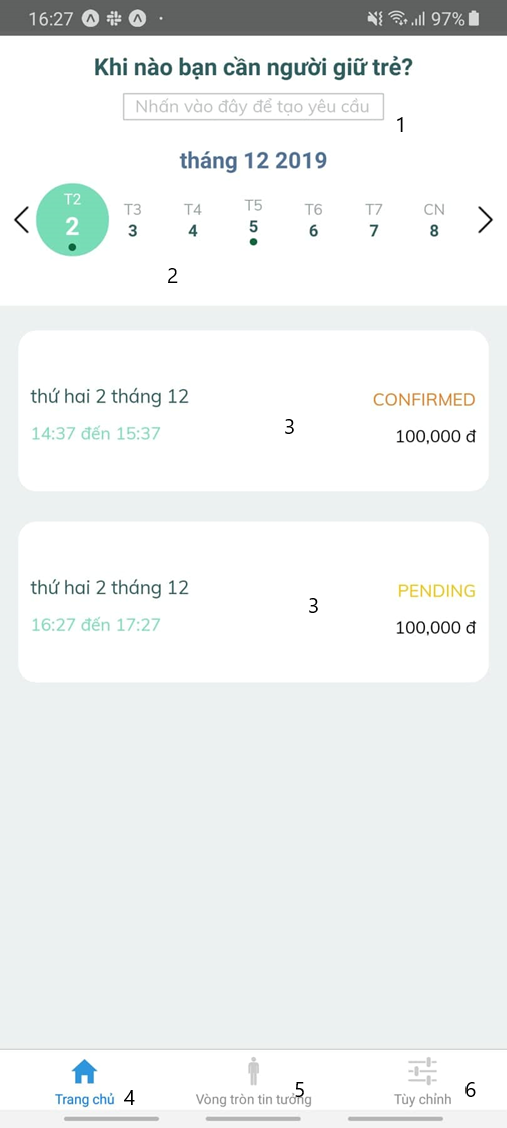
2

3

6

5

4



**Buttons/Hyperlink**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No** | **Function** | **Description** | **Validation** | **Outcome** |
| 1 | createRequest | Create request | N/A | Transfer to Create Request Screen |
| 2 | selectDayRequest | Change day | N/A | Day choose will have green color and dot under day choose if this day have request |
| 3 | request | Navigate to next screen | N/A | Transfer to Request Detail Screen |
| 4 | navigate | Navigate to next screen | N/A | Transfer to Home Screen |
| 5 | navigate | Navigate to next screen | N/A | Transfer to Circle Screen |
| 6 | navigate | Navigate to next screen | N/A | Transfer to Setting Screen |

##### Create request screen

7

6

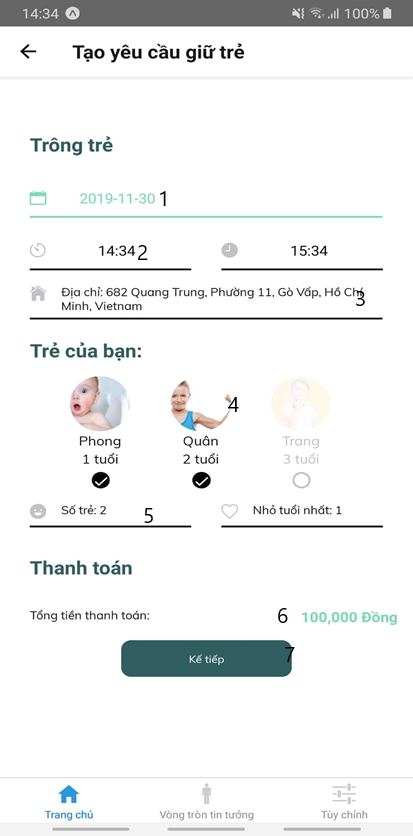
5

4

3

2

1



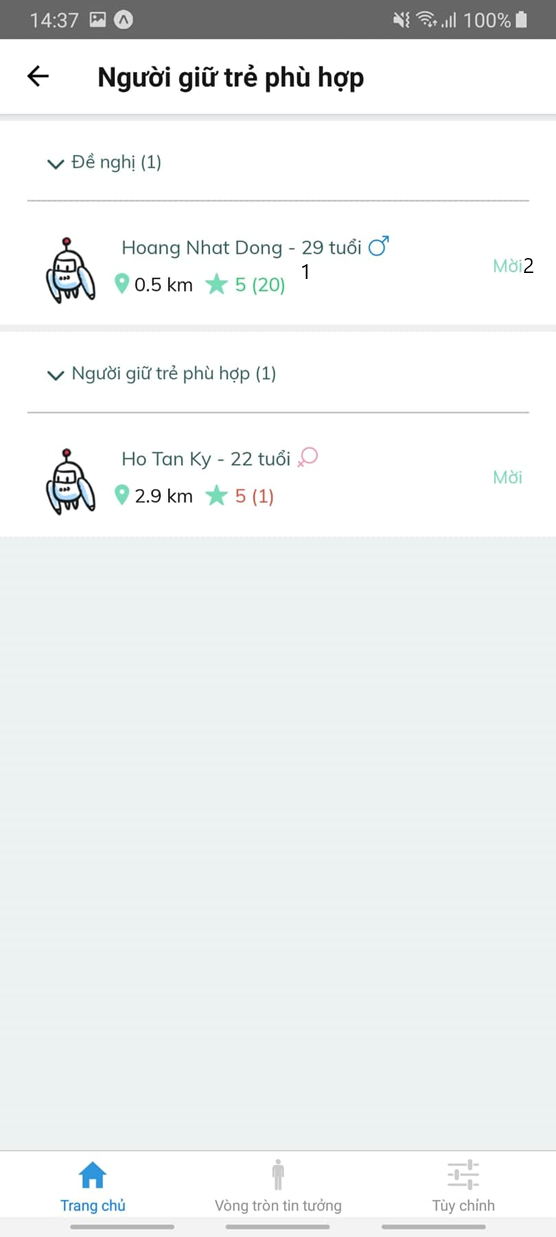
**Fields**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **No** | **Field Name** | **Description** | | **Read only** | | **Mandatory** | | **Control type** | | **Data type** | | **Length** | |
| 1 | chooseDay | | Choose day to create request | | No | | Yes | | Day Picker | | datetime | | N/A | |
| 2 | chooseTime | | Choose time to create request | | No | | Yes | | Day Picker | | datetime | | N/A | |
| 3 | parentAddress | | Show address of parent | | Yes | | Yes | | Input Text | | Datetime | | N/A | |
| 4 | selectChild | | Choose children to create request | | No | | Yes | | Image and radio button | | Boolean | | N/A | |
| 5 | showChild | | Show number and age of children | | No | | Yes | | Number | | Number | | N/A | |
| 6 | inputMoney | | Show money when create request | | Yes | | Yes | | Number | | Number | | N/A | |

**Buttons/Hyperlink**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No** | **Function** | **Description** | **Validation** | **Outcome** |
| 7 | recommend | Recommend babysitter | N/A | Transfer to Recommend screen |

##### Recommend screen



2

1

**Buttons/Hyperlink**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No** | **Function** | **Description** | **Validation** | **Outcome** |
| 1 | sitterProfile | Profile detail of babysitter | N/A | Transfer to Babysitter Profile screen |
| 2 | sendInvitation | Invite babysitter after create bank card | N/A | Input bank popup display |

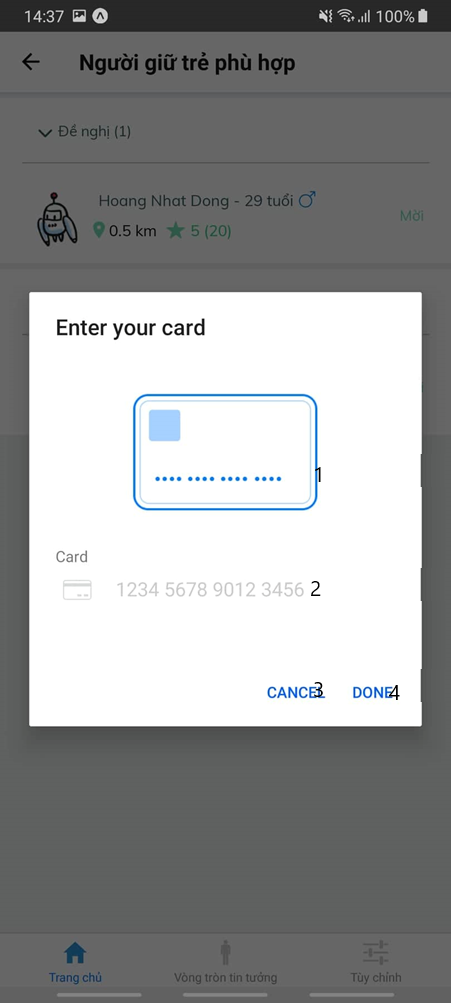
##### Bank card popup

1

4

3

2



**Fields**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **No** | **Field Name** | **Description** | **Read only** | **Mandatory** | **Control type** | **Data type** | **Length** |
| 1 | cardDisplay | Show bank card input | Yes | Yes | number | number | 16 |
| 2 | inpuCard | Input card number | No | Yes | Input number | number | 16 |

**Buttons/Hyperlink**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No** | **Function** | **Description** | **Validation** | **Outcome** |
| 3 | cancelInputBank | Cancel input bank card | N/A | Back to Recommend screen |
| 4 | doneInputBank | Done input bank card and create request | N/A | Back to Recommend screen with change button “Mời” to “Đã mời” |

3

1

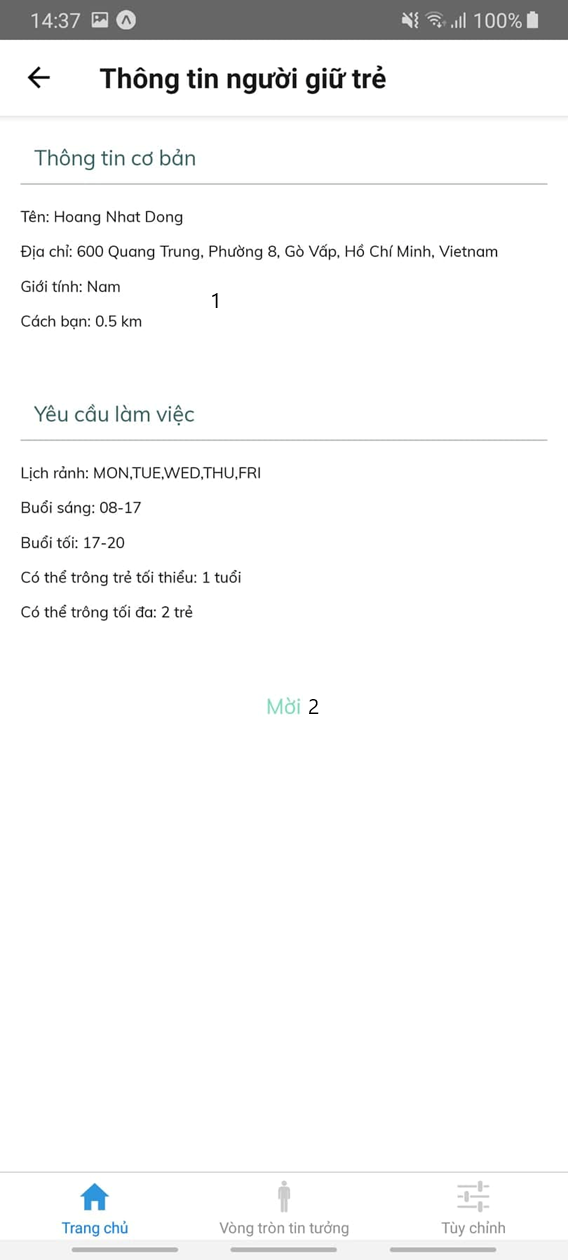
2

##### Profile Sitter Recommend Detail

1

2

1



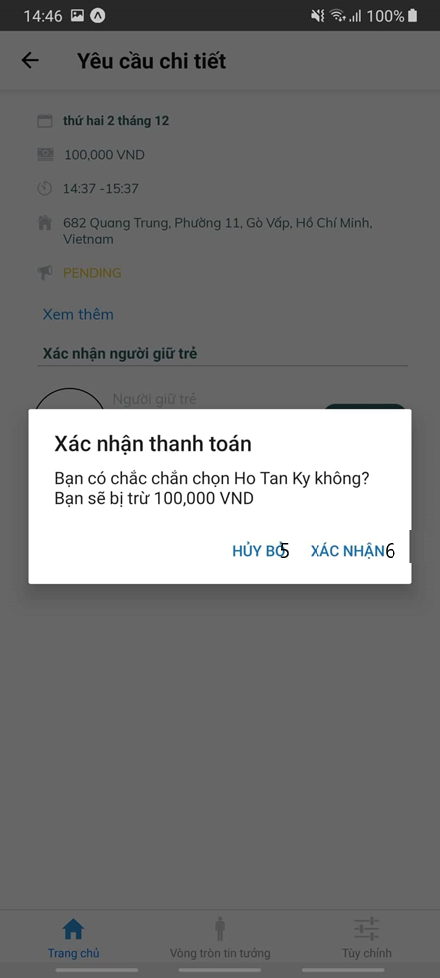
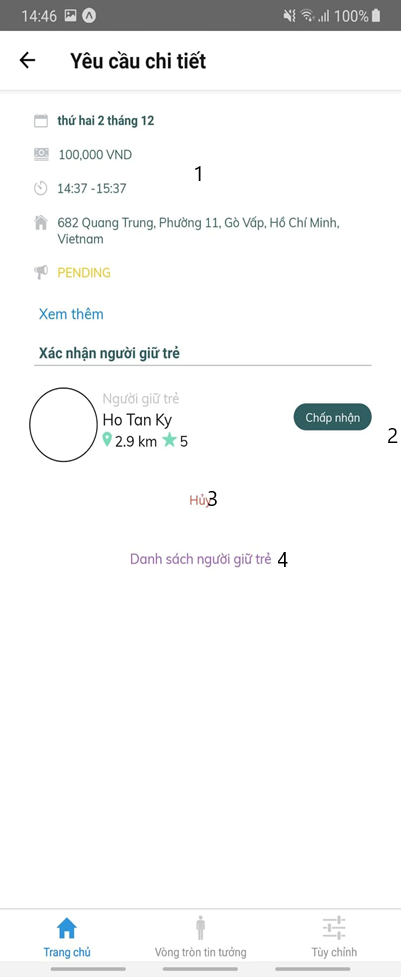
|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **No** | | **Field Name** | **Description** | **Read only** | **Mandatory** | **Control type** | **Data type** | **Length** |
| 1 | viewDetailSitter | | View information about sitter | Yes | Yes | text | text | N/A |

**Fields**

**Buttons/Hyperlink**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No** | **Function** | **Description** | **Validation** | **Outcome** |
| 2 | inviteSitter | Invite sitter | N/A | Change button “Mời” to “Đã mời” |

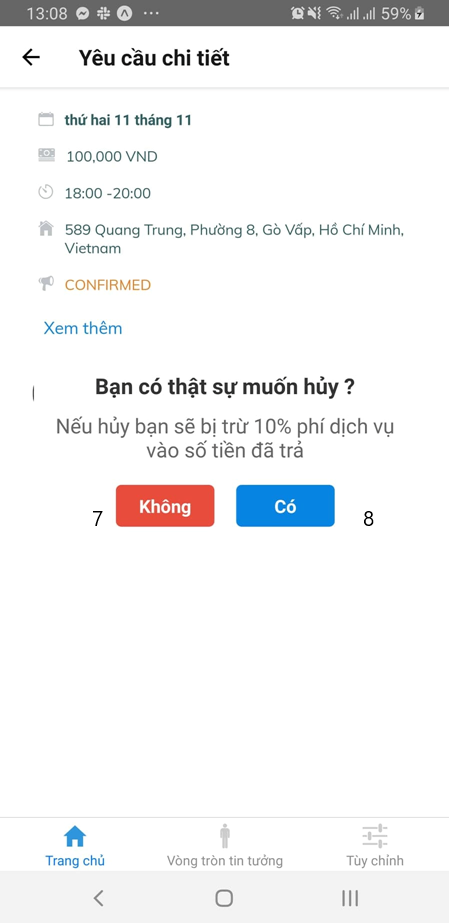
##### Request Detail



6

5

7



**Fields**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **No** | | **Field Name** | **Description** | **Read only** | **Mandatory** | **Control type** | **Data type** | **Length** |
| 1 | requestDetail | | Display request detail | Yes | Yes | Text and daytime | Text, daytime | N/A |

**Buttons/Hyperlink**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No** | **Function** | **Description** | **Validation** | **Outcome** |
| 2 | inviteSitter | Cancel input bank card | N/A | Back to Recommend screen |
| 3 | cancelRequest | View cancel request popup | N/A | Display cancel request popup |
| 4 | displayRecommend | View recommend sitter | N/A | Transfer to recommend screen |
| 5 | cancelInvite | Cancel invite popup | N/A | Hide popup |
| 6 | inviteSitter | Send request to sitter | N/A | Back to request detail screen with change babysitter hired |
| 7 | cancelPopup | Do nothing | N/A | Back to request detail screen |
| 8 | repay | Repay parent when cancel request | N/A | Transfer to home screen and repay parent 10% money when cancel request |

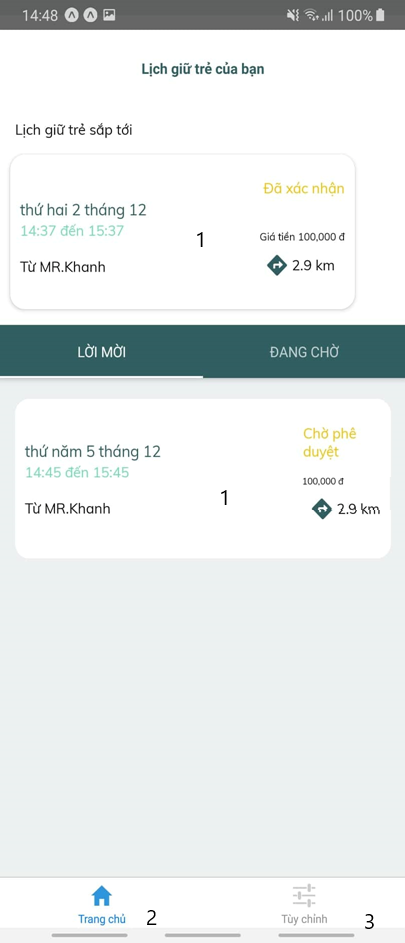
#### Sitter interface design

##### Sitter home screen

3

2

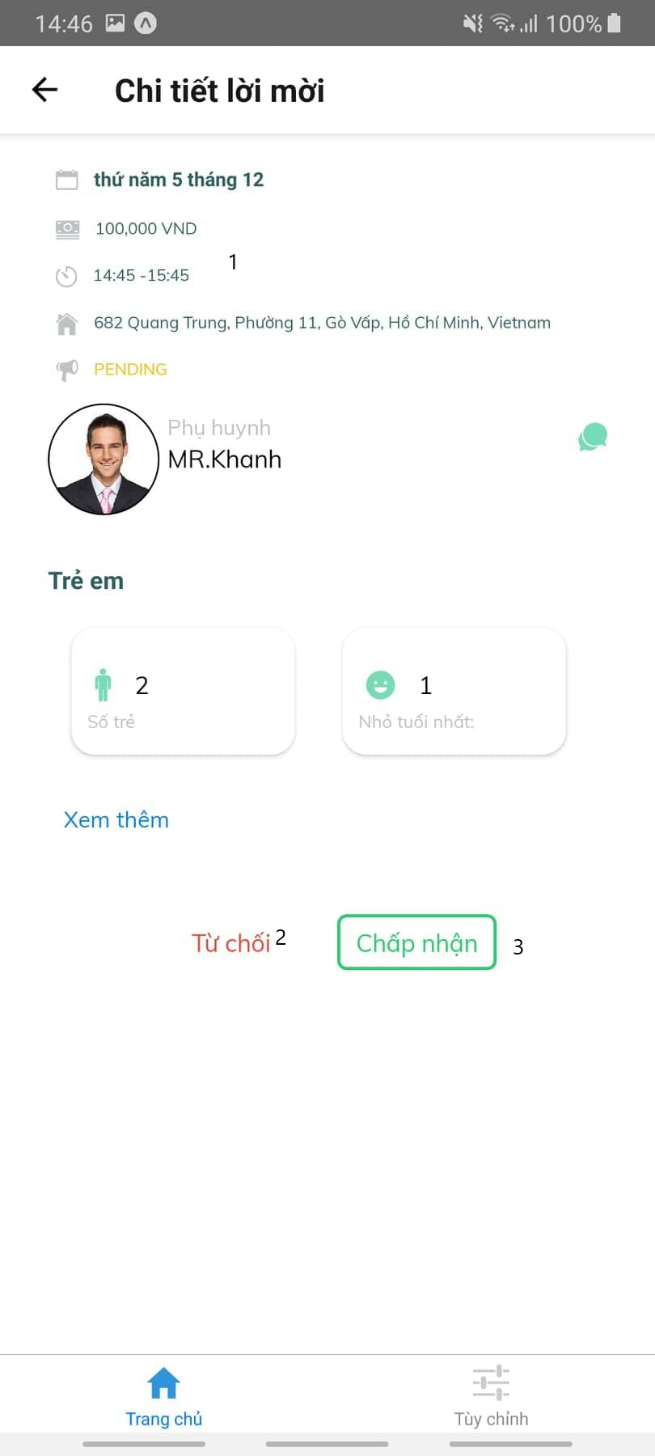
1



**Buttons/Hyperlink**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No** | **Function** | **Description** | **Validation** | **Outcome** |
| 1 | invitation | Navigate to next screen | N/A | Transfer to Invitation Detail Screen |
| 2 | navigate | Navigate to next screen | N/A | Transfer to Home Screen |
| 3 | navigate | Navigate to next screen | N/A | Transfer to Setting Screen |

##### Invitation Detail



**Fields**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **No** | | **Field Name** | **Description** | **Read only** | **Mandatory** | **Control type** | **Data type** | **Length** |
| 1 | invitationDetail | | Display invitation detail | Yes | Yes | Text and daytime | Text, daytime | N/A |

**Buttons/Hyperlink**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No** | **Function** | **Description** | **Validation** | **Outcome** |
| 2 | cancelInvitation | Cancel invitation from parent | N/A | Back to Home screen and delete invitation |
| 3 | acceptInvitation | Accept invitation from parent | N/A | Back to Home screen and change status invitation to “Chờ phê duyệt” |

##### QR code Babysitter



**Fields**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **No** | | **Field Name** | **Description** | **Read only** | **Mandatory** | **Control type** | **Data type** | **Length** |
| 1 | QR Code | | Display Qr code for parent to scan when babysitter checkin and checkout | Yes | Yes | Input image | stream | N/A |

## Database Design

### Entity relationship diagram (ERD)

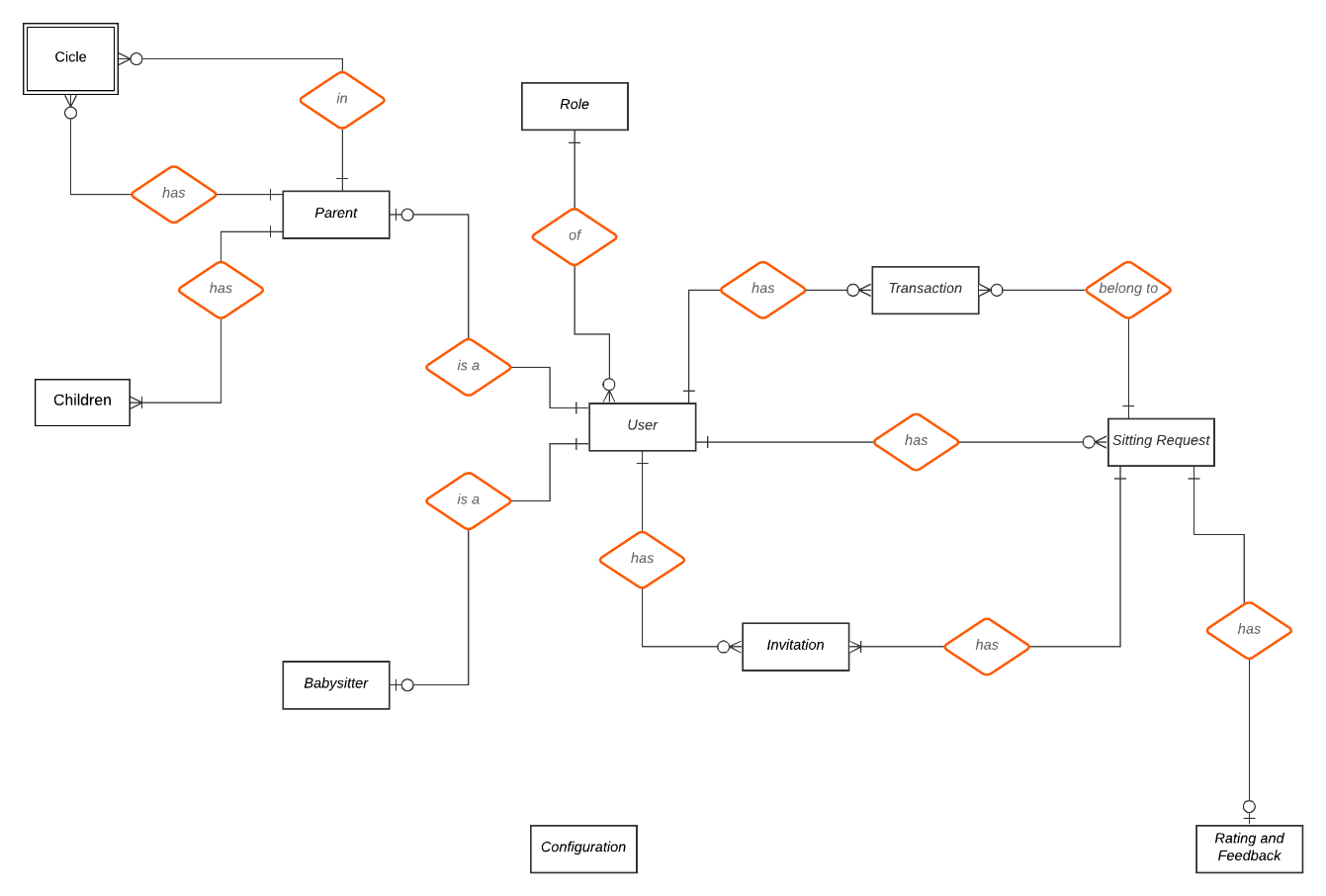


Figure 45 – Entity relationship diagram (ERD)

### Data Dictionary

|  |  |
| --- | --- |
| **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 66 - Data Dictionary

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Entity Name** | **Attributes** | **Description** | **Domain** | **Null** |
| User | id {PK} | Primary Key | INT(11) | N |
| phoneNumber {UX} | Phone Number | VARCHAR(255) | N |
| email {UX} | Email | VARCHAR(255) | N |
| password | Password | VARCHAR(255) | N |
| nickname | Name | VARCHAR(255) | N |
| gender | Gender | ENUM(…) | N |
| dateOfBirth | Birth day | DATE | N |
| address | Address | VARCHAR(255) | N |
| active | Account is active or not | TINYINT | N |
| createdAt | Created At | DATETIME | N |
| updatedAt | Update At | DATETIME | N |
| roleId {FK} | Role | INT(11) | N |

Table 67- User

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Entity Name** | **Attributes** | **Description** | **Domain** | **Null** |
| Role | id {PK} | Primary Key | INT(11) | N |
| roleName {UX} | Role Name | VARCHAR(255) | N |

Table 68 - Role

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Entity Name** | **Attributes** | **Description** | **Domain** | **Null** |
| Parent | userId {PK} | Primary Key | INT(11) | N |
| familyDescription | Family description | VARCHAR(255) | Y |
| parentCode {UX} | Parent Code | VARCHAR(255) | Y |
| createdAt | Created At | DATETIME | N |
| updatedAt | Update At | DATETIME | N |

Table 69 - Parent

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Entity Name** | **Attributes** | **Description** | **Domain** | **Null** |
| Babysitter | userId {PK} | Primary Key | INT(11) | N |
| weeklySchedule | Weekly Schedule | VARCHAR(255) | Y |
| daytime | Daytime | VARCHAR(255) | Y |
| evening | Evening | VARCHAR(255) | Y |
| minAgeOfChildren | Min Age Of Children | INT(11) | N |
| maxNumOfChildren | Max Number of Children | INT(11) | N |
| maxTravelDistance | Max Travel Distance | INT(11) | N |
| averageRating | Average Rating | FLOAT | Y |
| totalFeedBack | Total Feedback | INT(11) | Y |
| createdAt | Created At | DATETIME | N |
| updatedAt | Update At | DATETIME | N |

Table 70 - Babysitter

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Entity Name** | **Attributes** | **Description** | **Domain** | **Null** |
| Children | id {PK} | Primary Key | INT(11) | N |
| name | Name | VARCHAR(255) | Y |
| age | Age | INT(11) | N |
| Image | Image | TEXT | Y |
| parentId {FK} | Parent Id | INT(11) | N |
| createdAt | Created At | DATETIME | N |
| updatedAt | Update At | DATETIME | N |

Table 71 - Children

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Entity Name** | **Attributes** | **Description** | **Domain** | **Null** |
| Circle | id {PK} | Primary Key | INT(11) | N |
| ownerId | Owner Id | INT(11) | N |
| friendId | Friend Id | INT(11) | N |
| createdAt | Created At | DATETIME | N |
| updatedAt | Update At | DATETIME | N |

Table 72 - Circle

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Entity Name** | **Attributes** | **Description** | **Domain** | **Null** |
| Sitting Request | id {PK} | Primary Key | INT(11) | N |
| acceptedBabysitter {FK} | Weekly Schedule | INT(11) | Y |
| childrenNumber | Daytime | INT(11) | Y |
| minAgeOfChildren | Min Age Of Children | INT(11) | Y |
| sittingDate | Sitting Date | DATE | N |
| startTime | Start Time | TIME | N |
| endTime | End Time | TIME | N |
| sittingAddress | Sitting Address | VARCHAR(255) | N |
| totalPrice | Total Price | INT(11) | Y |
| status | Status | ENUM(…) | N |
| distance | Distance | VARCHAR(255) | Y |
| createdUser {FK} | Created User | INT(11) | N |
| createdAt | Created At | DATETIME | N |
| updatedAt | Update At | DATETIME | N |

Table 73 - Sitting Request

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Entity Name** | **Attributes** | **Description** | **Domain** | **Null** |
| Rating and Feedback | requestId {PK} {FK} | Primary Key | INT(11) | N |
| rating | Rating | INT(11) | Y |
| description | Description | VARCHAR(255) | Y |
| createdAt | Created At | DATETIME | N |
| updatedAt | Update At | DATETIME | N |

Table 74 - Rating and Feedback

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Entity Name** | **Attributes** | **Description** | **Domain** | **Null** |
| Transaction | id {PK} | Primary Key | INT(11) | N |
| chargeId | Charge Id | VARCHAR(255) | N |
| type | Type | ENUM(…) | N |
| description | Description | VARCHAR(255) | Y |
| amount | Amount | FLOAT | Y |
| requestId {FK} | Request Id | INT(11) | N |
| userId {FK} | User Id | INT(11) | N |
| createdAt | Created At | DATETIME | N |
| updatedAt | Update At | DATETIME | N |

Table 75 - Transaction

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Entity Name** | **Attributes** | **Description** | **Domain** | **Null** |
| Configuration | id {PK} | Primary Key | INT(11) | N |
| Date | Weekly Schedule | DATE | Y |
| startTime | startTime | TIME | Y |
| endTime | endTime | TIME | Y |
| description | description | VARCHAR(255) | Y |
| price | price | INT(11) | Y |
| remindBeforeDuration\_0 | Reminder time 1st | INT(11) | Y |
| remindBeforeDuration\_1 | Reminder time 2nd | INT(11) | Y |
| checkinTimeout | Check-in Timeout | INT(11) | Y |
| checkouTimeout | Checkout Timeout | INT(11) | Y |
| timezone | Timezone | VARCHAR(255) | Y |
| maxTravelDistance | Max Travel Distance | INT(11) | Y |
| circleWeight | Circle Weight | FLOAT | Y |
| ratingWeight | Rating | FLOAT |  |
| distanceWeight | Distance | FLOAT | Y |
| minimumFeedback | Minimum Feedback | INT(11) | Y |
| createdAt | Created At | DATETIME | N |
| updatedAt | Update At | DATETIME | N |

Table 76 - Configuration

# System implementation and Testing

## Introduction

### Overview

This section describes the approach and methodologies used by group to plan, organize and manage the testing of BID system. It provides in the detail all necessary information about the implementation and testing procedure of the system included test plans, test cases, test result, test environments, pass/fail criteria and risks estimations as well as a checklist to cover all possible cases.

### Test Approach

* Goal: Test all features in the whole BID system based on the core flow.
* Method: black-box testing
* Technique: check list

The testing for this project will consists of Integration System test level. Testing the program which was integrated and as a complete system to ensure that the software requirements have been met.

* Integration testing would be performed by all member of team and approved by team leader.
* System testing is focused on assessing the system’s reliability. This process is concerned with finding errors that result from unanticipated interactions between components and component interface problems.

## Database relationship Diagram

### Physical Diagram

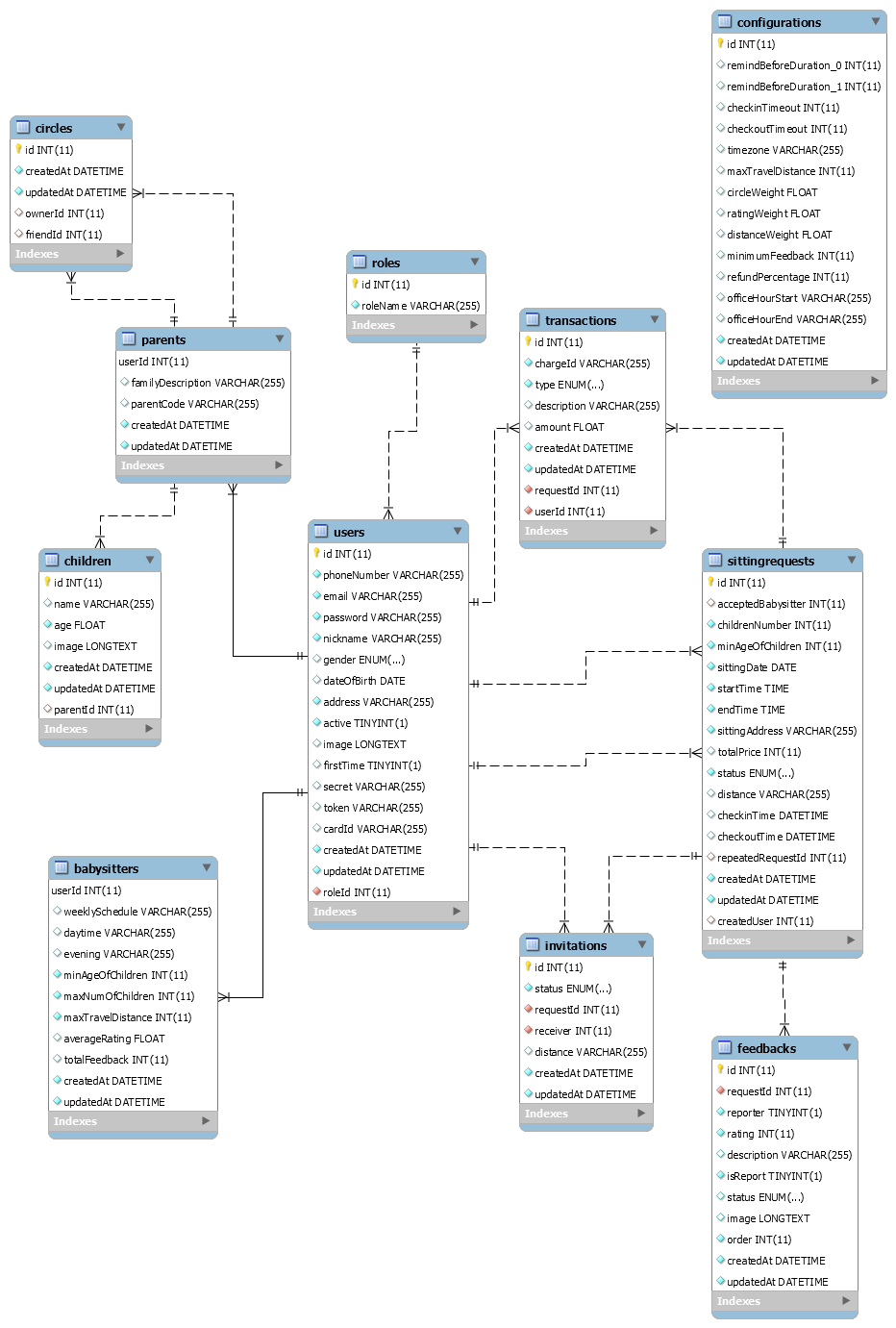


Figure 46 - Physical Diagram

### Data Dictionary

|  |  |
| --- | --- |
| 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 77 - Entity Data Dictionary

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Entity Name** | **Attributes** | **Description** | **Domain** | **Null** |
| User | id {PK} | Primary Key | INT(11) | N |
| phoneNumber {UX} | Phone Number | VARCHAR(255) | N |
| email {UX} | Email | VARCHAR(255) | N |
| password | Password | VARCHAR(255) | N |
| nickname | Name | VARCHAR(255) | N |
| gender | Gender | ENUM(…) | N |
| dateOfBirth | Birth day | DATE | N |
| address | Address | VARCHAR(255) | N |
| active | Account is active or not | TINYINT | N |
| createdAt | Created At | DATETIME | N |
| updatedAt | Update At | DATETIME | N |
| roleId {FK} | Role | INT(11) | N |

Table 78 - User

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Entity Name** | **Attributes** | **Description** | **Domain** | **Null** |
| Role | id {PK} | Primary Key | INT(11) | N |
| roleName {UX} | Role Name | VARCHAR(255) | N |

Table 79 - Role

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Entity Name** | **Attributes** | **Description** | **Domain** | **Null** |
| Parent | userId {PK} | Primary Key | INT(11) | N |
| familyDescription | Family description | VARCHAR(255) | Y |
| parentCode {UX} | Parent Code | VARCHAR(255) | Y |
| createdAt | Created At | DATETIME | N |
| updatedAt | Update At | DATETIME | N |

Table 80 - Parent

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Entity Name** | **Attributes** | **Description** | **Domain** | **Null** |
| Babysitter | userId {PK} | Primary Key | INT(11) | N |
| weeklySchedule | Weekly Schedule | VARCHAR(255) | Y |
| daytime | Daytime | VARCHAR(255) | Y |
| evening | Evening | VARCHAR(255) | Y |
| minAgeOfChildren | Min Age Of Children | INT(11) | N |
| maxNumOfChildren | Max Number of Children | INT(11) | N |
| maxTravelDistance | Max Travel Distance | INT(11) | N |
| averageRating | Average Rating | FLOAT | Y |
| totalFeedBack | Total Feedback | INT(11) | Y |
| createdAt | Created At | DATETIME | N |
| updatedAt | Update At | DATETIME | N |

Table 81 - Parent

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Entity Name** | **Attributes** | **Description** | **Domain** | **Null** |
| Children | id {PK} | Primary Key | INT(11) | N |
| name | Name | VARCHAR(255) | Y |
| age | Age | INT(11) | N |
| Image | Image | TEXT | Y |
| parentId {FK} | Parent Id | INT(11) | N |
| createdAt | Created At | DATETIME | N |
| updatedAt | Update At | DATETIME | N |

Table 82 - Children

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Entity Name** | **Attributes** | **Description** | **Domain** | **Null** |
| Circle | id {PK} | Primary Key | INT(11) | N |
| ownerId | Owner Id | INT(11) | N |
| friendId | Friend Id | INT(11) | N |
| createdAt | Created At | DATETIME | N |
| updatedAt | Update At | DATETIME | N |

Table 83 - Circle

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Entity Name** | **Attributes** | **Description** | **Domain** | **Null** |
| Sitting Request | id {PK} | Primary Key | INT(11) | N |
| acceptedBabysitter {FK} | Weekly Schedule | INT(11) | Y |
| childrenNumber | Daytime | INT(11) | Y |
| minAgeOfChildren | Min Age Of Children | INT(11) | Y |
| sittingDate | Sitting Date | DATE | N |
| startTime | Start Time | TIME | N |
| endTime | End Time | TIME | N |
| sittingAddress | Sitting Address | VARCHAR(255) | N |
| totalPrice | Total Price | INT(11) | Y |
| status | Status | ENUM(…) | N |
| distance | Distance | VARCHAR(255) | Y |
| createdUser {FK} | Created User | INT(11) | N |
| createdAt | Created At | DATETIME | N |
| updatedAt | Update At | DATETIME | N |

Table 84 - Sitting Request

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Entity Name** | **Attributes** | **Description** | **Domain** | **Null** |
| Rating and Feedback | requestId {PK} {FK} | Primary Key | INT(11) | N |
| rating | Rating | INT(11) | Y |
| description | Description | VARCHAR(255) | Y |
| createdAt | Created At | DATETIME | N |
| updatedAt | Update At | DATETIME | N |

Table 85 - Rating and Feedback

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Entity Name** | **Attributes** | **Description** | **Domain** | **Null** |
| Transaction | id {PK} | Primary Key | INT(11) | N |
| chargeId | Charge Id | VARCHAR(255) | N |
| type | Type | ENUM(…) | N |
| description | Description | VARCHAR(255) | Y |
| amount | Amount | FLOAT | Y |
| requestId {FK} | Request Id | INT(11) | N |
| userId {FK} | User Id | INT(11) | N |
| createdAt | Created At | DATETIME | N |
| updatedAt | Update At | DATETIME | N |

Table 86 - Transaction

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Entity Name** | **Attributes** | **Description** | **Domain** | **Null** |
| Configuration | id {PK} | Primary Key | INT(11) | N |
| Date | Weekly Schedule | DATE | Y |
| startTime | startTime | TIME | Y |
| endTime | endTime | TIME | Y |
| description | description | VARCHAR(255) | Y |
| price | price | INT(11) | Y |
| remindBeforeDuration\_0 | Reminder time 1st | INT(11) | Y |
| remindBeforeDuration\_1 | Reminder time 2nd | INT(11) | Y |
| checkinTimeout | Check-in Timeout | INT(11) | Y |
| checkouTimeout | Checkout Timeout | INT(11) | Y |
| timezone | Timezone | VARCHAR(255) | Y |
| maxTravelDistance | Max Travel Distance | INT(11) | Y |
| circleWeight | Circle Weight | FLOAT | Y |
| ratingWeight | Rating | FLOAT |  |
| distanceWeight | Distance | FLOAT | Y |
| minimumFeedback | Minimum Feedback | INT(11) | Y |
| createdAt | Created At | DATETIME | N |
| updatedAt | Update At | DATETIME | N |

Table 87 - Configuration

## Performance Measures

### Web application page load speed

#### Definition

This section tests the general page load speed from all the page of the web application.

#### Test environment

**Server:**

* **Operating system:** Ubuntu 14.04 Server 32 bit
* **RAM:** 1024 MB
* **Storage:** 60 GB
* **Processor:** 2.6 GHz Intel Core i5
* **Network:** 2 Mbps

#### Test cases

Using browser from client side and go to each page of the web application. Record the page load time using Chrome Dev Tools at client side.

**Test pages:**

|  |  |
| --- | --- |
| Staff | Dashboard  - Create acount for parent and sitter  - Edit configuration |
| Request:   * list request * detail request * detail request * edit request |
| Invitation:   * list all invitation * list invitation by id * detail invitation * edit invitation |

Table 88 - Test pages web application

#### Test result

The test is run 10 times, each time we go through all the pages listed above and log down the average page load time.

|  |  |  |
| --- | --- | --- |
| Test No. | Average page load time (second) | Execute date |
| 1 | 1.1 | 10 May 2019 |
| 2 | 1.0 | 10 May 2019 |
| 3 | 1.35 | 10 May 2019 |
| 4 | 1.59 | 10 May 2019 |
| 5 | 2.0 | 10 May 2019 |
| 6 | 0.95 | 10 May 2019 |
| 7 | 2.12 | 10 May 2019 |
| 8 | 1.2 | 10 May 2019 |
| 9 | 1.54 | 10 May 2019 |
| 10 | 1.63 | 10 May 2019 |
|  | **Average: 1.45 (seconds)** |  |

Table 89 - Web application page load speed test result

### Mobile application API load speed

#### Definition

This section tests the load speed of mobile apps when connection to server through API.

#### Test environment

**Server:**

* **Operating system:** Ubuntu 14.04 Server 32 bit
* **RAM:** 1024 MB
* **Storage:** 60 GB
* **Processor:** 2.6 GHz Intel Core i5
* **Network:** 2 Mbps

**Client:**

|  |  |  |
| --- | --- | --- |
| Hardware Requirements | Minimum | Recommended |
| Internet Connection | 512Kbps | Wi-Fi Connection 12MB |
| Operating System | Android 6.0 | Android 7.0 |
| Memory | 128MB of RAM | 1GB of RAM or more |

Table 90 - Mobile app API load speed client specification

#### Test cases

Using mobile application to send the entire API request to server.

List of APIs:

|  |  |  |
| --- | --- | --- |
| No. | API | Description |
| 1 | getRequestList | Get request list by parent |
| 2 | getRequestDetail | Get request detail for parent |
| 3 | getRecommendList | Get recommend list after create request |
| 4 | getInformationDetail | Get information detail parent and sitter |
| 5 | getHistoryList | Get history list by parent and sitter |
| 6 | getInvitationList | Get invitation list by sitter |
| 7 | getInvitationDetail | Get invitation detail for sitter |
| 8 | getReferference | Get referference for sitter to change |

Table 91 - Mobile app API load speed test cases

#### Test result

The test is run 10 times, each time we run all the test cases listed above and log down the average API response time.

|  |  |  |
| --- | --- | --- |
| Test No. | Average page load time (second) | Execute date |
| 1 | 2.13 | 13 May 2019 |
| 2 | 2.89 | 13 May 2019 |
| 3 | 2.21 | 13 May 2019 |
| 4 | 2.5 | 13 May 2019 |
| 5 | 2.87 | 13 May 2019 |
| 6 | 1.73 | 13 May 2019 |
| 7 | 2.17 | 13 May 2019 |
| 8 | 1.58 | 13 May 2019 |
| 9 | 1.62 | 13 May 2019 |
| 10 | 1.31 | 13 May 2019 |
|  | **Average: 2.10 (seconds)** |  |

Table 92 - Mobile application API load speed test result

## Test Plan

The overall purpose of testing is to ensure MIC system meets its entire technical, functional and business requirement. The purpose of this document is to describe the overall test plan and strategy for testing the MIC system. The approach described in this document provides the framework for all testing related to this application. Individual test cases will be written for each version of the application that is released. This document will also be updated as required for each release.

### Features to be tested

* Answer invitation
* Create babysitting request
* Accept babysitter
* Payment: Pay
* Recommend babysitter to parent

### Features not to be tested

* Admin
* Staff
* Send feedback
* Login
* Logout
* Get profile
* Edit sitting preferences
* Get invitation list
* Get invitation
* Get sitting history
* Save transaction record
* Generate identification token
* Edit configuration
* Create user account
* Edit user information
* Ban user
* Get babysitting request list
* Get babysitting request
* Get request history
* Add people to trust circle

## System testing test case

### Communication Diagrams

Test cases are created from bellow communication diagram:

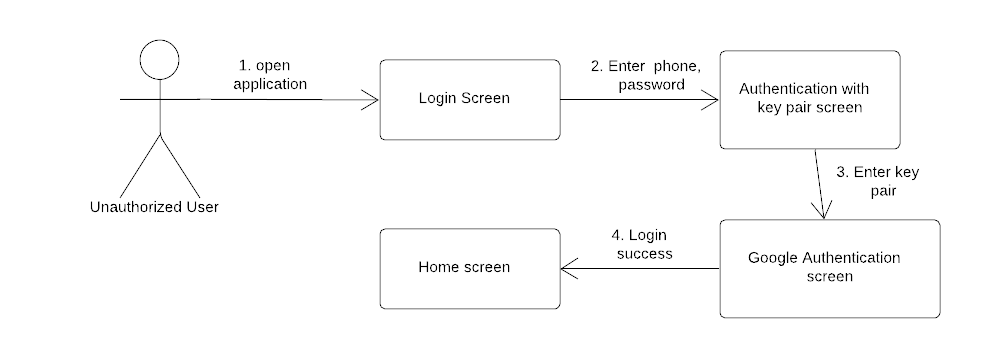


Figure 47 - <Unauthorized User> authentication flow

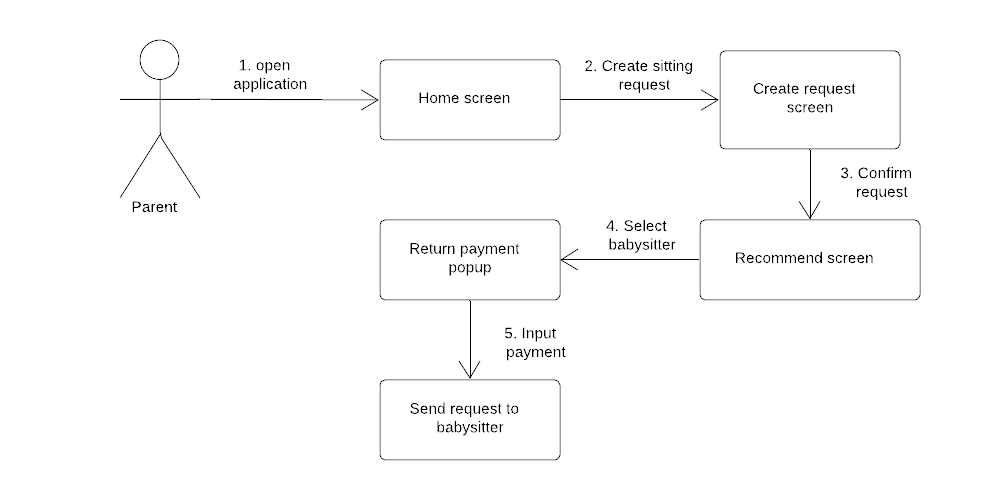


Figure 48 - <Parent> Create request flow

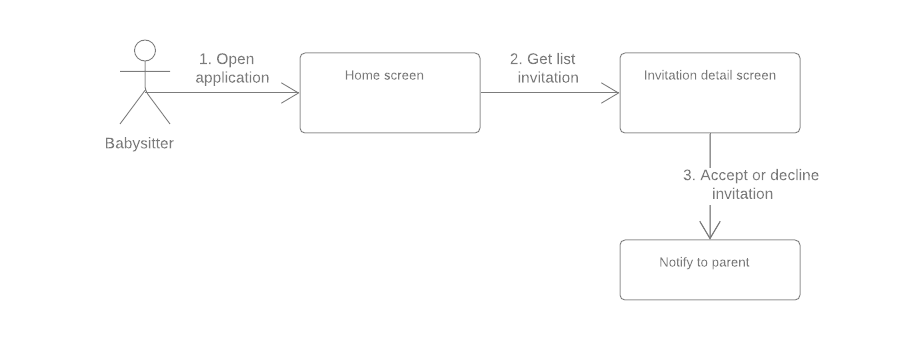


Figure 49 - <Sitter> Answer invitation flow

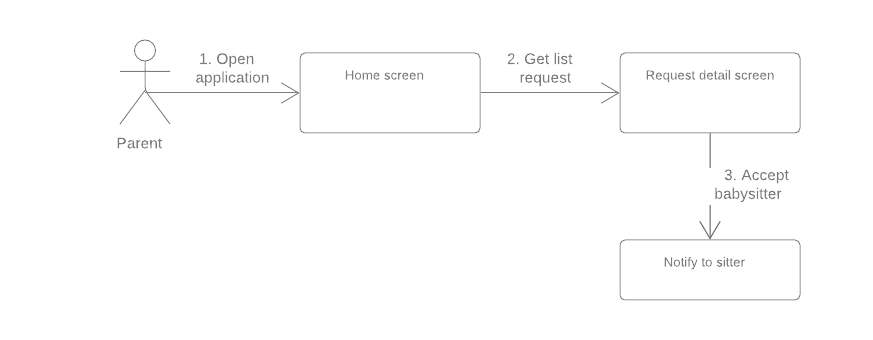


Figure 50 - <Parent> Accept babysitter

### Test case

#### <Parent> Create request

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Test Case Description | Test case procedure | Expected output | Inter-test case dependence | Result |
| CR1 | Test create new request with valid information | 1. Login with account name “Mr Khanh” with role “Parent”  2. Go to the home page  3. Click “Nhấn vào đây để tạo yêu cầu”  4. Enter request informationl  5. Click "Kế tiếp" button. | Go to recommend page |  | Pass |
| CR2 | Test create new request with duplicate information | 1. Create new request with information like test case "CR1”. | Show message:"Bạn đang có yêu cầu giữ trẻ trong khung giờ này, đặt thêm sẽ mất tiền, bạn có muốn tiếp tục" | CR1 | Pass |
| CR3 | Test create new request with wrong date (date and time in the past) | 1. Create new request with information like test case "CR1" but change request information  2. Change request information | Date and time in the past are disabled | CR1 | Pass |
| CR4 | Test create new request with wrong time (Start time is greater than end time) | 1. Create new request with information like test case "CR1" but change request information  2. Change request information | Show error message: “Bạn không thể chọn giờ bắt đầu lớn hơn giờ kết thúc” | CR1 | Pass |
| CR5 | Test create new request with with no child selected | 1. Create new request with information like test case "CR1" but change request information | Show error message: “Bạn phải chọn ít nhất một trẻ để tạo yêu cầu” | CR1 | Pass |
| CR6 | Test create new request with with no day selected | 1. Create new request with information like test case "CR1" but change request information | Show error message: “Bạn phải chọn ngày để tạo yêu cầu” | CR1 | Pass |
| CR7 | Test create new request with with no time selected | 1. Create new request with information like test case "CR1" but change request information | Show error message: “Bạn phải chọn giờ để tạo yêu cầu” | CR1 | Pass |
| CR8 | Test cancel request | 1. Create new request with information like test case "CR1" and send request successful with case “PM1”  2. Click “Hủy” button | Show message” Bạn có muốn hủy yêu cầu này, bạn sẽ bị trừ 10% chi phí bạn đã thuê” | CR1, PM1 | Pass |

Table 93 - Test case <Parent> Create request

#### <Parent> Payment

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Test Case Description | Test case procedure | Expected output | Inter-test case dependence | Result |
| PM1 | Test input bank card in new request with valid information | 1. Register new contract with information like test case "CR1".  2. Display input bank card popup  3. Enter bank card information  4. Click "Done" button | Show input bank card popup and create request successful, send invitation to babysitters with name “Ho Tan Ky” and wait for the babysitter to respond. | CR1 | Pass |
| PM2 | Test input bank card in new request with invalid information | 1. Register new contract with information like test case "CR1"  2. Display input bank card popup  3. Enter wrong bank card information  4. Click "Done" button | Show error message: “Bạn nhập sai thông tin, mời nhập lại” | CR1 | Pass |

Table 94 - Test case <Parent> Payment

#### <Sitter> Answer invitation

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Test Case Description | Test case procedure | Expected output | Inter-test case dependence | Result |
| AI1 | Test accept invitation from parent | 1. Login with account “Ho Tan Ky” with role “Sitter”  2. Go to home screen  3. Choose invitation information from test case “CR1”  4. Go to invitation detail  5. Click “Chấp nhận” button | System change status invitation to “Đang chờ phê duyệt” | CR1 | Pass |
| AI2 | Test cancel invitation from parent | 1. Login with account “Ho Tan Ky” with role “Sitter”  2. Go to home screen  3. Choose invitation information from test case “CR1”  4. Go to invitation detail  5. Click “Từ chối” button | System change status invitation to “Đã hủy” and save to history | CR1 | Pass |

Table 95 - Test case <Sitter> Accept invitation

#### <Parent > Accept babsitter

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Test Case Description | Test case procedure | Expected output | Inter-test case dependence | Result |
| AB1 | Test accept babysitter | 1. Login with account “Mr Khanh” with role “Parent”  2. Request was created successfully from test case “CR1, PM1” and show name babysitter accepted by test case “AI1”  3. Go to home screen  4. Go to request detail  5. Click “Chấp nhận” button | System change status request and invite status to “Đã xác nhận” | CR1, PM1, AI1 | Pass |
| AB2 | Test accept babysitter at the same time (Two parents accept the same babysitter) | 1. Login with account “Duong chi dai” with role “Parent”  2. Request was created successfully from test case “CR1, PM1” but change account “Duong chi dai” and show name babysitter accepted by test case “AI1”  3. Go to home screen  4. Go to request detail  5. Click “Chấp nhận” button at the same time with account “Mr Khanh” | One of account will receive a message error “Yêu cầu đã bị trùng”, the other will change status request to “Đã xác nhận” | CR1, PM1, AI1 | Pass |

Table 96 - Test case <Parent> Accept babysitter

#### <Parent> Recommend babysitter

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Test Case Description | Test case procedure | Expected output | Inter-test case dependence | Result |
| RB1 | Test recommend babysitter | 1. Login with account “Mr Khanh” with role “Parent”  2. Request was created successfully from test case “CR1”  3. Navigate to recommend screen | Show babysitter fit the request | CR1 | Pass |
| RB2 | Test recommend babysitter fail | 1. Login with account “Mr Khanh” with role “Parent”  2. Request was created successfully from test case “CR1” but change day  3. Navigate to recommend screen | A message "Không tìm thấy người giữ trẻ nào phù hợp với lịch của bạn" is shown | CR1 | Pass |

Table 97 - <Parent> Recommend

### Test case results statistics

|  |  |  |  |
| --- | --- | --- | --- |
| Test case | Runs | Pass rate (%) | Day test |
| CR1 | 10 | 98 | 4/4/2019 |
| CR2 | 10 | 100 | 4/4/2019 |
| CR3 | 10 | 98 | 4/4/2019 |
| CR4 | 10 | 100 | 4/4/2019 |
| CR5 | 10 | 100 | 4/4/2019 |
| CR6 | 10 | 100 | 4/4/2019 |
| CR7 | 10 | 98 | 5/4/2019 |
| CR8 | 10 | 100 | 5/4/2019 |
| PM1 | 10 | 96 | 5/4/2019 |
| PM2 | 10 | 100 | 5/4/2019 |
| AI1 | 10 | 100 | 8/4/2019 |
| AI2 | 10 | 100 | 8/4/2019 |
| AB1 | 10 | 100 | 8/4/2019 |
| AB2 | 10 | 98 | 8/4/2019 |
| RB1 | 10 | 100 | 8/4/2019 |
| RB2 | 10 | 100 | 8/4/2019 |

Table 98 - Test case results statistics

### 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.

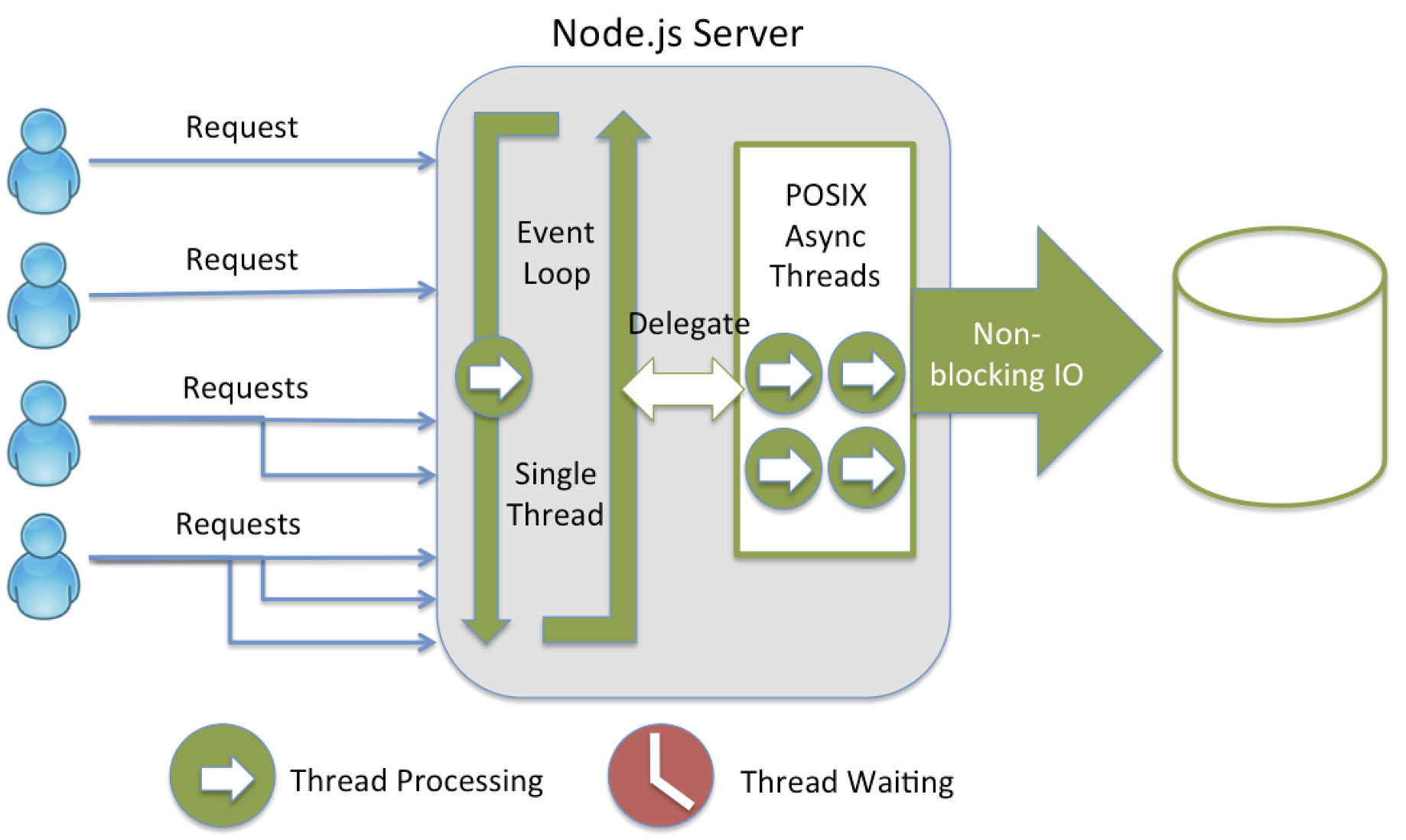


Figure 51 – NodeJS server(2)

#### Socket.io

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

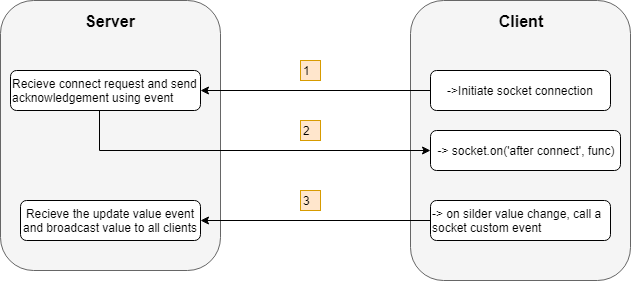


Figure 52 – Web. Socket communication with browser(4)

### Mobile Framework: React Native

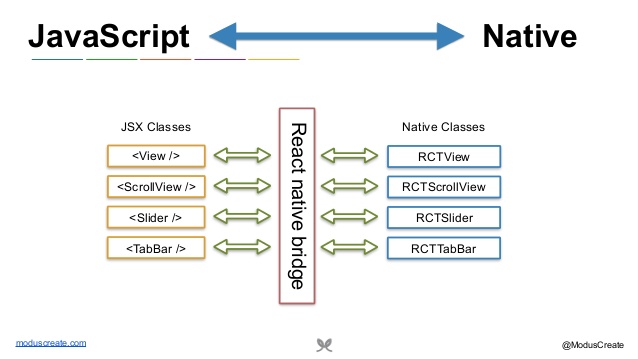


Figure 56 - React native bridge(3)

#### 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.

# Software User’s Manual

## Installation Guide

### **Setting up environment at server side**

#### **Hardware requirements**

|  |  |  |
| --- | --- | --- |
| **Windows** | **Minimum Requirement** | **Recommended** |
| **Internet Connection** | Cable, Wi-Fi (6 Mbps) | Cable, Wi-Fi (16 Mbps) |
| **Operating System** | Ubuntu 18.04 LTS or above | Ubuntu 18.04 LTS or above |
| **Computer Processor** | Core i7 (5th) | Core i7 (7th) |
| **Computer Memory** | 8GB RAM | 16GB RAM |

Table 99 - Hardware requirements

#### **Software requirements**

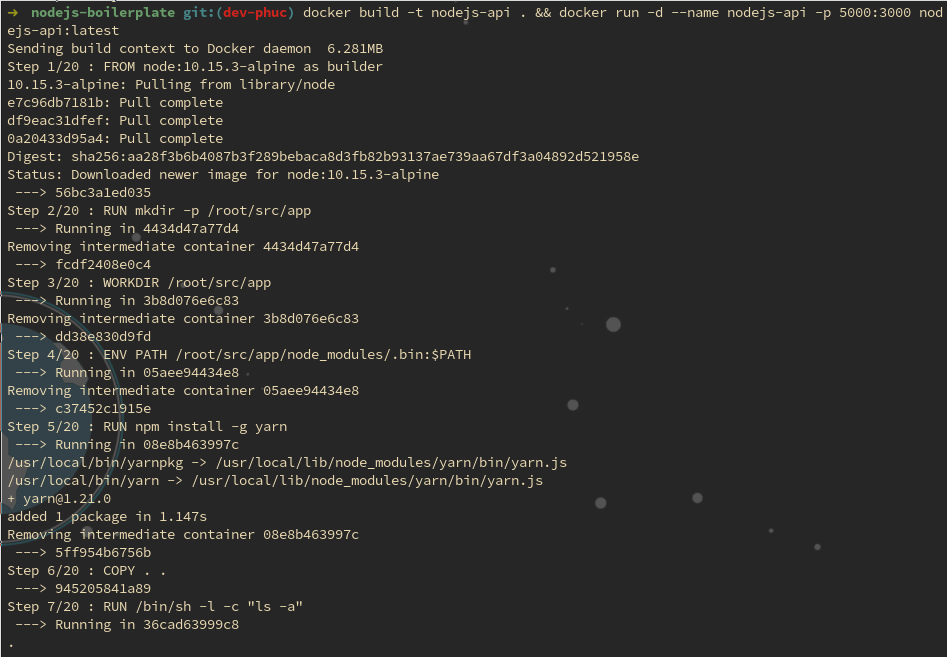
|  |  |  |
| --- | --- | --- |
| **Software** | **Name / Version** | **Description** |
| **Operating system** | Ubuntu 18.04, Window 10 Pro or MacOS 10+ | Operating system and platform for development/production |
| **Environment** | Docker version 18.09 | Containerization environment |
| **Modeling tool** | Lucid Chart, Adobe Xd | Used to implement diagrams and prototypes |
| **IDE** | Visual Studio Code | Visual Studio IDE |
| **DBMS** | MySQL Database | MySQL database |
| **Source control** | GitHub, (Gittea) | Gittea (self hosted private source control) |
| **API Testing** | Postman or Insomnia | Testing APIs |
| **Web browser** | Chrome, Firefox, Safari, Edge | Support for multiple platform. |

Table 100 - Software requirements

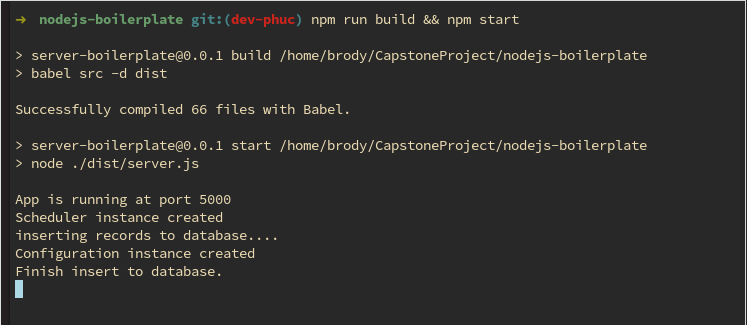
### **Deployment at Production server**

Setup the Docker container

* Clone the repository: git clone [**https://git.babysitinde.live/capstone-project/nodejs-boilerplate.git**](https://git.babysitinde.live/capstone-project/nodejs-boilerplate.git)
* Switch to the repository context: cd nodejs-boilerplate
* Deploy the Docker in your os with command: docker build -t nodejs-api . && docker run -d --name nodejs-api -p 5000:3000 nodejs-api:latest



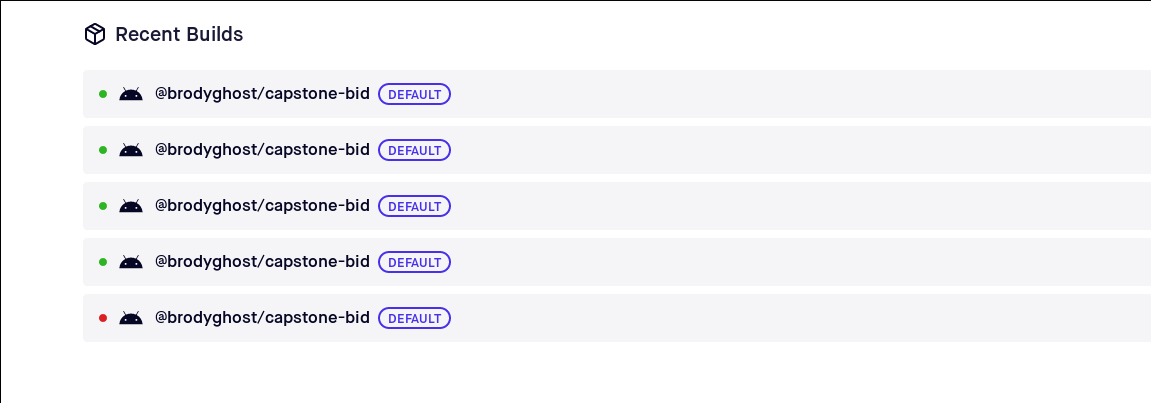
* Or Build production and run without using container: npm run build && npm start



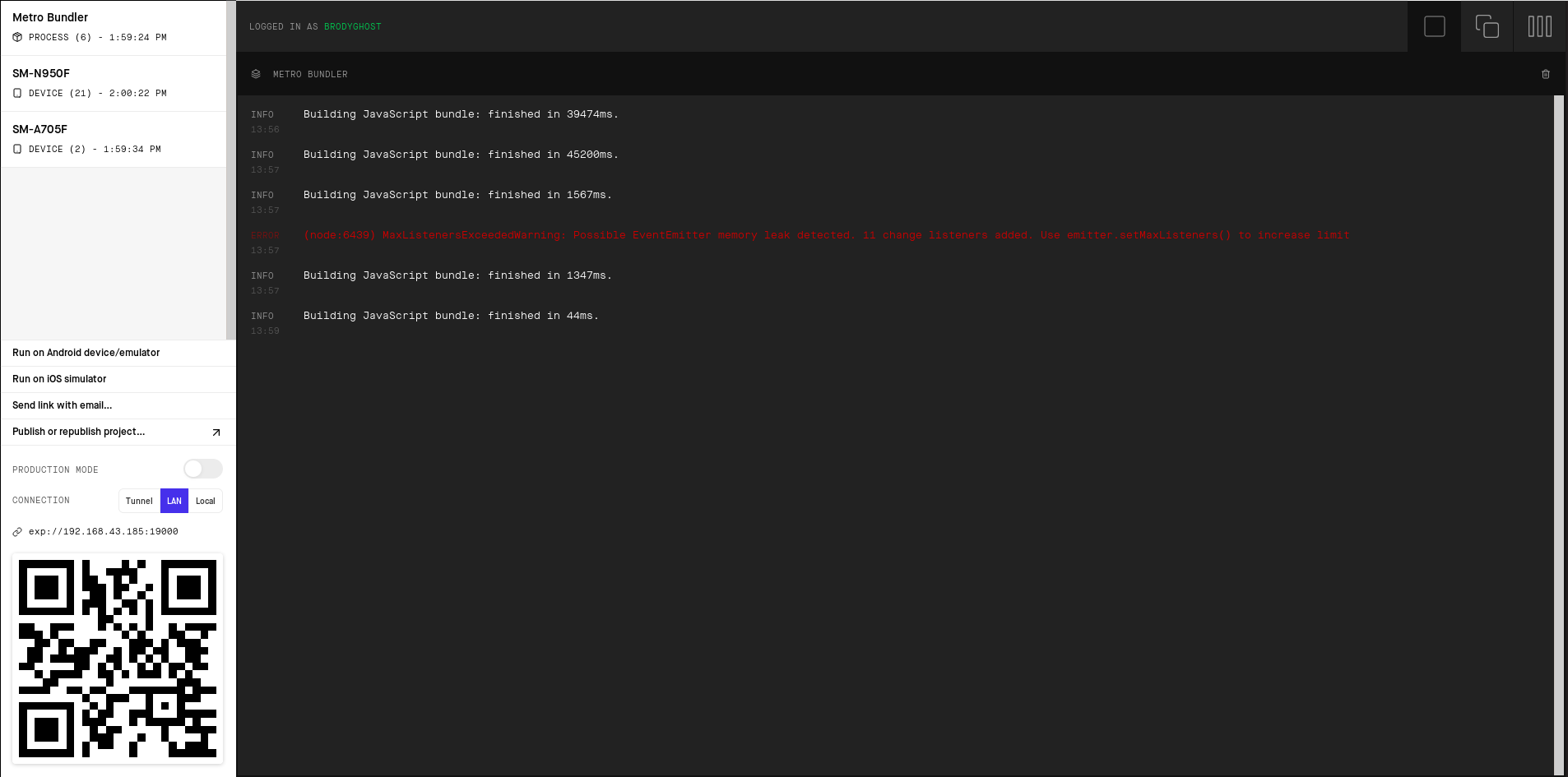
### **Setting up the environment at client side**

* Use our Builds apk and pod file to install to your device

[**https://expo.io/dashboard/brodyghost/builds/ae251e76-a0bb-47de-97d7-51962b33c691**](https://expo.io/dashboard/brodyghost/builds/ae251e76-a0bb-47de-97d7-51962b33c691)



* Or Clone the repository: **git clone** [**https://git.babysitinde.live/capstone-project/react-native-boilerplate.git**](https://git.babysitinde.live/capstone-project/react-native-boilerplate.git)
* Switch to the repository context: **cd react-native-boilerplate**
* Install Expo client: **npm install –g expo-cli**
* Start the development server: **npm start or yarn start**



* Scan the QR code to run application on your mobile

# Appendix

**Reference**:

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

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

**[2]** All About Node.Js, Expressjs 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/ **>**